Designing with TTL Integrated Circuits

TEXAS INSTRUMENTS ELECTRONICS SERIES

VISIT…

LANZAROTE Caliente.COM

Designing with
TTL Integrated Circuits

TEXAS INSTRUMENTS ELECTRONICS SERIES

Crawford	MOSFET IN CIRCUIT DESIGN
Delhom	DESIGN AND APPLICATION OF TRANSISTOR SWITCHING CIRCUITS
The Engineering Staff of Texas Instruments Incorporated	CIRCUIT DESIGN FOR AUDIO, AM/FM, AND TV
The Engineering Staff of Texas Instruments Incorporated	SOLID-STATE COMMUNICATIONS
The Engineering Staff of Texas Instruments Incorporated	TRANSISTOR CIRCUIT DESIGN
The IC Applications Staff of Texas Instruments Incorporated	DESIGNING WITH TTL INTEGRATED CIRCUITS
Hibberd	INTEGRATED CIRCUITS
Hibberd	SOLID-STATE ELECTRONICS
Kane and Larrabee	CHARACTERIZATION OF SEMICONDUCTOR MATERIALS
Runyan	SILICON SEMICONDUCTOR TECHNOLOGY
Sevin	FIELD-EFFECT TRANSISTORS

Designing with TTL Integrated Circuits

Prepared by the IC Applications Staff of
Texas Instruments Incorporated

Edited by
Robert L. Morris and John R. Miller

Contributors

W. D. Anderson
A. G. Douce
R. C. Grimes
W. R. Heniford
R. L. Morris
R. F. Schweitzer
S. Wolff

McGRAW-HILL BOOK COMPANY

New York St. Louis San Francisco Düsseldorf Johannesburg
Kuala Lumpur London Mexico Montreal New Delhi
Panama Rio de Janeiro Singapore Sydney Toronto

Sponsoring Editor Tyler G. Hicks
Director of Production Stephen J. Boldish
Editing Supervisor Linda B. Hander
Editing and Production Staff Gretlyn Blau,
Teresa F. Leaden, George E. Oechsner

DESIGNING WITH TTL INTEGRATED CIRCUITS

Copyright © 1971 by Texas Instruments Incorporated. All Rights Reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Texas Instruments Incorporated. *Library of Congress Catalog Card Number* 77-150465

07-063745-8

6 7 8 9 0 HDBP 7 5 4 3

Preface

During every new era in the history of a technology, there emerges a class of devices so versatile, economical, and reliable that they become known as the workhorses. In this integrated-circuit era of logic-system technology, TTL (transistor-transistor logic) integrated circuits have clearly achieved that eminence.

This book is devoted exclusively to TTL circuits. It will familiarize the reader with the entire TTL family, their basic descriptions, electrical performance, and applications. Since the intelligent selection of TTL circuits depends upon sound digital-system concepts and logic-design techniques, much of the book lays this foundation of theory. Throughout the book, however, emphasis is on the *practical* implementation of logic functions. To this end, a great many working circuits are presented as examples. And to this end, the authors, relying on their applications experience, have tried to anticipate and solve the problems that most often face the practical designer.

An approach has been used that makes this book valuable not only to the logic-systems designer but to a broad spectrum of readers—from hobbyists to designers of powerful computers, and from electronics companies to industries now far-removed from electronics. Obviously, one book cannot encompass all knowledge in this complex field, nor can it avoid becoming outdated in some particulars regarding products; both of these problems are easily solved, however, by contacting Texas Instruments.

Information contained in this book is believed to be accurate and reliable. However, responsibility is assumed neither for its use nor for any infringement of patents or rights of others which may result from its use. No license is granted by implication or otherwise under any patent or patent right of Texas Instruments or others.

Texas Instruments Incorporated
Components Group

Contents

Preface v

Chapter 1. Introduction to Digital Logic **1**

1.1 The Binary System 1
1.2 Logic Circuits 2
1.3 Use of Logic Gates 4
1.4 Logic Designation 8
1.5 Comparison of Logic Types 8
1.6 Levels of Integrated-circuit Complexity 13

Chapter 2. Series 54/74 Overview **15**

2.1 Typical Characteristics 15
2.2 Standard Series 54/74 TTL 16
2.3 Series 54/74 Low-power TTL 18
2.4 Series 54H/74H High-speed TTL 18
2.5 Series 54S/74S Schottky-clamped TTL 21

Chapter 3. Circuit Analysis and Characteristics of Series 54/74 . . . . **23**

3.1 Basic TTL Operation 23
3.2 TTL Advantages 24
3.3 Circuit Parameters 25
3.4 Circuit Characteristics of Specialized Gates 45
APPENDIX TO CHAPTER 3: *TTL Loading Rules* 57

Chapter 4. Extended-range Operation **67**

4.1 Integrated-circuit Components 67
4.2 Device Reaction to External Influences 69
4.3 Voltage Breakdown for Input and Output 79

Chapter 5. Noise Considerations **83**

5.1 Noise Types and Control Methods 84
5.2 Shielding 84
5.3 Grounding and Decoupling 85
5.4 Cross Talk 93
5.5 Transmission-line Reflections 96

Chapter 6. Combinational Logic Design **107**

6.1 Basic Functions 107
6.2 Postulates and Theorems 108
6.3 Logic Expressions 109
6.4 Simplification and Minimization 110
6.5 Practical Logic Gates 114
6.6 Analysis of Logic Circuits 115
6.7 Implementation of Logic Expressions 118
6.8 Implementation of Fundamental Logic Functions 119
6.9 Combinational Logic Applications 131
Bibliography 159

Chapter 7. Flip-flops **161**

7.1 Flip-flop Types 163
7.2 Series 54/74 Flip-flops 171
7.3 Flip-flop Applications 173

Chapter 8. Decoders **181**

8.1 Decoder Theory 181
8.2 Series 54/74 Decoders and Decoder/Drivers 189
8.3 Applications of Decoders 202

Chapter 9. Arithmetic Elements **211**

9.1 Addition of Binary Numbers 211
9.2 Parallel Binary Addition 213
9.3 Serial Binary Adder 214
9.4 Series 54/74 TTL Arithmetic Elements 214
9.5 Binary Representations for Computer Arithmetic 220
9.6 Addition and Subtraction of Decimal Numbers with Binary Representations 224
9.7 Fast Binary Addition 235
9.8 Adder Applications to Binary Number Representation Conversion 240

Chapter 10. Counters **243**

10.1 Ripple Counters 243
10.2 Synchronous Counters 248
10.3 Series 54/74 Counters 254
10.4 Counter Implementation and Applications 271

Chapter 11. Shift Registers **285**

11.1 SN54/74 Shift Registers 286
11.2 Shift Register Counters and Generators 292
11.3 Other Shift Register Applications 305

Chapter 12. Other Applications **309**

12.1 A Simple Binary Multiplier 309
12.2 12-hour Digital Clock 313
12.3 Serial Gray Code to Binary Conversion 313
12.4 Modulo-360 Adder 315

Index 319

1

Introduction to Digital Logic

The first electronic computers were very cumbersome because they used the decimal system, which required 10 distinct levels for each order. The problem of defining and maintaining these 10 levels proved to be so great that the decimal system was replaced by a simple binary system with only two levels or digits (0 and 1). In binary arithmetic, a quantity either exists or does not exist, and this type of decision making is relatively easy to implement with transistor circuits, where a voltage either exists or does not exist at the output. And since the transistor can change from one condition to the other in less than one-millionth of a second, it can make at least a million decisions per second.

The basic operation performed by digital-computer logic is the process of addition. Subtraction, multiplication, and division are carried out by modifications of the addition process. For example, to multiply 15 by 5, the digital computer simply adds five 15s. Although a great many binary circuits are required for such an operation, the simplicity of the system and the economics of the transistor and the integrated circuit have made binary digital computers extremely attractive.

Data are fed into a digital computer in the form of electrical pulses at two discrete voltage levels, and information is represented by the number of pulses and the time spacing between them. In most computers, data fed into the computer in decimal form are converted to equivalent binary form, and the arithmetic is carried out with binary numbers. The result is reconverted to decimal form at the output.

1.1 THE BINARY SYSTEM

An understanding of the operation of digital integrated circuits requires familiarity with the binary system and its use in logical decision making. In binary language, the first (or off) state is called 0, and the second (or on) state is called 1.

In the decimal system, we first count in units up to 9, and then for the next order we go back to unit 0 but insert a 1 in the second-order column to indicate that we have counted through all the units once. This gives us 10. To count with the binary scale, we follow exactly the same procedure, using only the numbers 0 and 1. After count 1, we have used all our units and must move to the second-order column

Table 1.1. Conversion from Decimal to Binary

Decimal	*Binary*
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000
32	100000
64	1000000
128	10000000

to indicate that we have counted through our scale once. Thus the number 2 in the decimal system is indicated by 10 (called *one-zero*, not *ten*) in the binary scale. The next count is indicated by changing the 0 to 1, giving 11 (*one-one*) corresponding to 3 in the decimal system. Now we have used all our units again, and so for the next count both columns must go back to 0, and we put a 1 in the third-order column, giving 100 (*one-zero-zero*) as the binary equivalent of 4.

Table 1.1 gives the equivalent binary numbers for some decimal numbers. Note that the binary number 10 is equal to decimal 2, which is 2^1; binary 100 equals decimal 4, or 2^2; binary 1000 equals decimal 8, or 2^3; and binary 10000 equals decimal 16, or 2^4. Every additional order of binary numbers corresponds to an additional power of 2. This fact is used in converting a binary number to its decimal equivalent. For example, take the binary number 11010. This is equivalent to $2^4 + 2^3 + 0 + 2^1 + 0$, or $16 + 8 + 0 + 2 + 0$, which equals 26. Conversely, a decimal number can be converted to binary by repeatedly subtracting the highest possible power of 2. Take the decimal number 26. First subtract 16, (2^4), which in binary is 10000. From the remaining 10, subtract 8, (2^3), or binary 1000. This leaves 2, or binary 10. Adding the binary numbers 10000 + 1000 + 10 gives 11010.

Binary numbers obviously require a longer sequence of digits than their decimal equivalents, especially the larger numbers; but because the electronic digital computer can process millions of simple additions per second, the relative size of binary numbers presents no serious problem.

1.2 LOGIC CIRCUITS

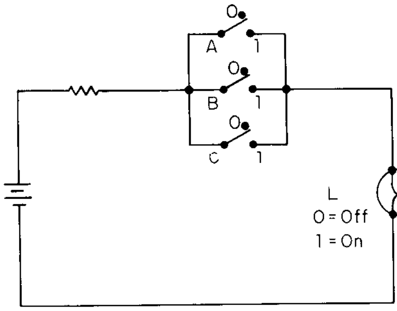
Logic circuits make the series of decisions necessary to obtain the logical answer to a problem having a given set of conditions. To make logic decisions, three basic logic circuits (called *gates*) are used: the OR circuit, the AND circuit, and the NOT circuit.

The OR Circuit. This basic circuit has two or more inputs and a single output. The inputs and the output can each be at one of two states, 0 or 1. The circuit is arranged so that the output is in state 1 when any one of the inputs is in state 1; i.e., the output is 1 when input A *or* input B *or* input C is 1. The circuit can be illustrated by the analogy shown in Fig. 1.1*a*. A battery supplies a lamp L through three switches in parallel. The switches are the three inputs to the lamp; the light from the lamp represents the circuit output.

If we define an open switch as a 0 state and no light as a 0 state, and we define a closed switch as a 1 state and a glowing lamp as a 1 state, we can list the various combinations of switch states and the resulting output states. This list is called a truth table and is shown in Fig. 1.1*b*. From the truth table it can be seen that *all* switches must be open (0 state) for the light to be off (output 0 state).

This type of circuit is called an OR gate and has the symbolic representation shown in Fig. 1.1*c*, which shows an OR gate with three inputs. Thus, the OR gate is used to make the logic decision on whether or not at least one of several inputs is in the 1 state.

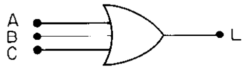
The AND Circuit. This circuit also has several inputs and only one output, but in this case the circuit output is at a logical 1 state only if *all* inputs are in the logical 1 state simultaneously. This is illustrated in Fig. 1.2*a*. Here lamp L lights only



(a)

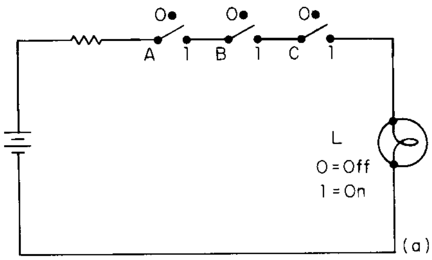
A	B	C	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b)



(c)

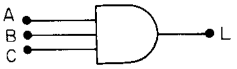
Fig. 1.1. The OR circuit: (*a*) analogy; (*b*) truth table; (*c*) symbol.



(a)

A	B	C	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b)



(c)

Fig. 1.2. The AND circuit: (*a*) analogy; (*b*) truth table; (*c*) symbol.

if switch *A,* switch *B, and* switch *C* are all closed at the same time. The lamp does not light if any one of the switches is open. With the same notation as before, the truth table for the AND circuit is shown in Fig. 1.2*b,* and the symbolic representation is shown in Fig. 1.2*c.* Thus, the AND gate makes the logic decision on whether or not several inputs are all in the 1 state at the same time.

This is a convenient spot for a brief digression into a description of fan-in and fan-out. The number of inputs to a gate is called the fan-in. In the above examples, each gate has a fan-in of three. There is only one output signal from a gate, but it may be required that this signal be fed to several other logic gates. The number of subsequent gates that the output of a particular gate can drive is called the fan-out.

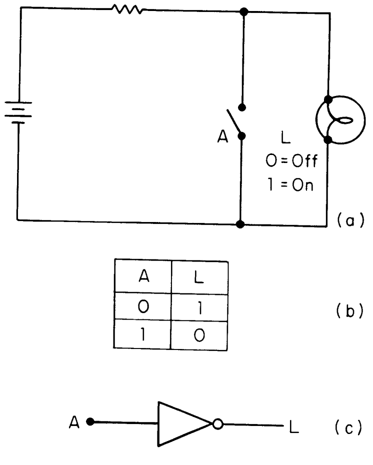
The NOT Circuit. This circuit has a single input and a single output and is arranged so that the output state is always opposite to the input state. Consider Fig. 1.3*a.* When the switch is open (0), current flows through the lamp and it lights (1). If the switch is closed, current flows through the switch rather than the lamp, which goes out. This operation of making the output state opposite to that of the input is called inversion, and a circuit designed to do this is called an inverter. Its simple truth table is shown in Fig. 1.3*b* and its symbol in Fig. 1.3*c.*

NOR and NAND Gates. A NOT circuit can be combined with an OR gate or an AND gate so that inversion occurs together with the gate function. A NOT circuit combined with an OR gate is called a NOR gate. This is illustrated using the lamp-circuit analogy in Fig. 1.4*a.* If any one of the switches is in the 1 state, the lamp is in the 0 state. The truth table is shown in Fig. 1.4*b* and the symbol in Fig. 1.4*c.*

Similarly, a NOT circuit combined with an AND gate is called a NAND gate (Fig. 1.5*a*). When all switches are in the 1 position, the lamp is in the 0 state. The truth table for the NAND circuit is shown in Fig. 1.5*b* and the symbol in Fig. 1.5*c.*

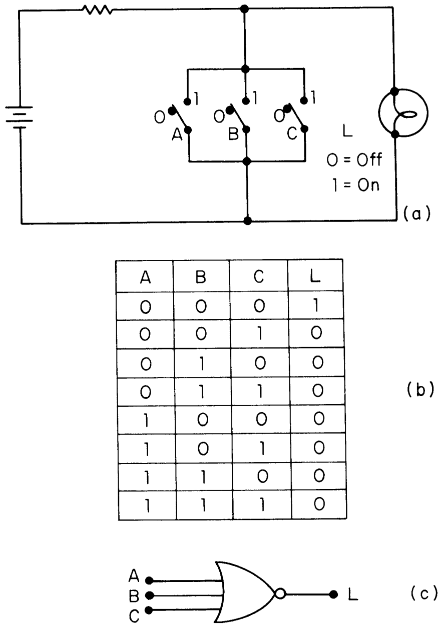
1.3 USE OF LOGIC GATES

To illustrate the use of logic gates, consider the operation of adding together two binary numbers, *A* and *B.* First, consider the simplest case, when *A* and *B* each



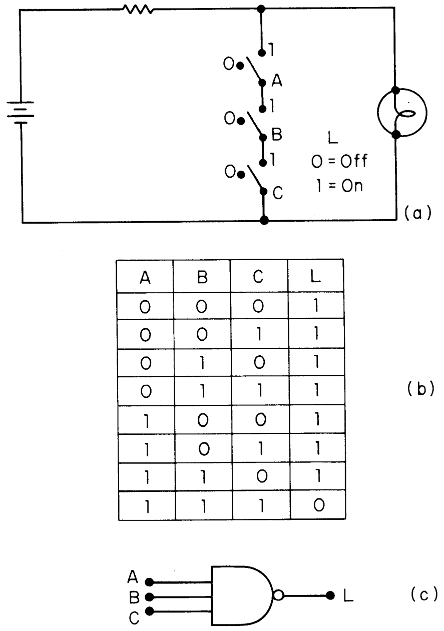
A	L
0	1
1	0

Fig. 1.3. The NOT circuit: (*a*) analogy; (*b*) truth table; (*c*) symbol.



A	B	C	L
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Fig. 1.4. The NOR circuit: (*a*) analogy; (*b*) truth table; (*c*) symbol.



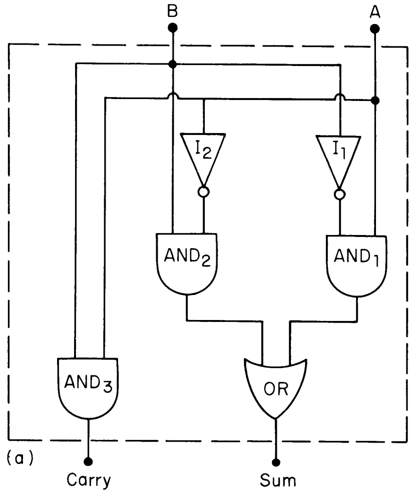
A	B	C	L
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Fig. 1.5. The NAND circuit: (*a*) analogy; (*b*) truth table; (*c*) symbol.

consist of one binary digit, either 0 or 1. The logic diagram of a circuit for this addition is shown in Fig. 1.6*a* and the overall truth table in Fig. 1.6*b*. The two inputs to the circuit, A and B, are each connected to an AND gate: A to AND$_1$, B to AND$_2$. A and B are also fed to the opposite gates through inverters I_1 and I_2. Thus when A is 1, its input to gate AND$_1$ is 1 and its input to AND$_2$ is 0; when A is 0, its input is 0 to AND$_1$ and 1 to AND$_2$. The outputs from the two AND gates are connected to an OR gate, and the output from the OR gate gives the sum S. A and B are also fed as direct inputs to a third AND gate AND$_3$, whose output gives the carry C.

Consider the operation of the circuit. There are four possible conditions:

1. $A = 0$ and $B = 0$. None of the AND gates can give an output since at least one input to each of them is 0. Thus both the sum and carry indicate 0, giving the answer 00.
2. $A = 1$ and $B = 0$. A gives a 1 to gate AND$_1$, and the 0 at B is inverted by I_1 to give another 1 to gate AND$_1$. Thus, both inputs to AND$_1$ are 1, and the gate operates to give a 1-output into the OR gate. Both inputs to gate AND$_2$ are 0, and so the output from this gate is 0. Since one of the inputs to the OR gate is 1, the output sum is 1. The carry gate AND$_3$ does not operate since one of its inputs is 0, and so the carry output is 0, giving the answer 01.
3. $A = 0$ and $B = 1$. The operation is the same as in condition 2 above, but inputs to AND$_1$ and AND$_2$ are reversed. Again, the answer is 01.
4. $A = 1$ and $B = 1$. Neither gate AND$_1$ nor AND$_2$ can give an output since



(b)

B	A	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

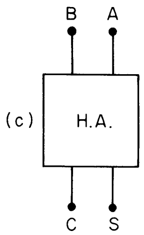
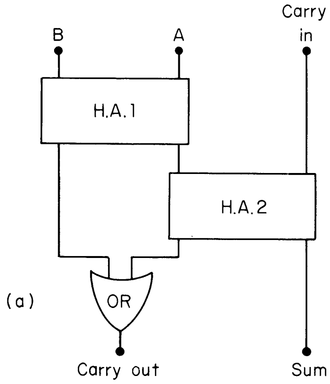


Fig. 1.6. The half-adder: (*a*) arrangement; (*b*) truth table; (*c*) symbol.



(b)

Carry in	B	A	Carry out	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Fig. 1.7. The full-adder: (*a*) arrangement; (*b*) truth table.

one of their inputs is 0 from the inverter, and so the sum is 0. But both inputs to the carry gate AND_3 are 1, and so it operates to give a carry output of 1. Thus, the answer is 10.

This circuit can be considered as a basic logic block (as shown by the dashed block in Fig 1.6) with two inputs and two outputs. As such it is called a *half-adder,* "half" because it adds only first-order numbers. When adding two numbers in an order higher than the first, it is necessary to make provision for the circuit to accept and add in a carry from the previous order. To do this, a full-adder circuit is necessary. One method of making a full-adder is to use two half-adders as in Fig. 1.7*a*. The first adds *A* and *B*, and the second adds the resulting sum to a carry input from the previous order to give the final sum. The carry outputs from the two half-adders are fed to an OR gate whose output gives the final carry. The truth table for the full-adder is given in Fig. 1.7*b*. A little analysis shows that a carry output cannot exist at the outputs of both half-adders simultaneously.

A series of full-adders in parallel can be used to add binary numbers of several orders. The first order will not have a carry input, and can be a half-adder. Figure 1.8*b* is the block diagram for a system to add together two 3-order binary numbers $A_3A_2A_1$ and $B_3B_2B_1$ as indicated in the addition table, Fig. 1.8*a*.

It is interesting to count the number of gates required for these adding units.

A single full-adder as in Fig. 1.7 needs six AND gates, three OR gates, and four inverters for a total of thirteen gates just to add two binary digits in the same order. Modern computers must handle about ten decimal orders—decimal numbers up to 10,000,000,000, or 2^{33}. In binary terms, this is 34 orders, or bits. Adding together two 34-bit binary numbers requires 33 full-adders and a half-adder, for a total of 435 gates.

Given the requirement of repeated addition for multiplication, and facilities for other manipulations, it is easy to see why the number of gates in the arithmetic unit of a modern digital computer can often exceed 10,000. In such a unit, there will be repeated use of the same type of gate; all the AND gates can be identical, and the same is true of all the OR gates and inverters. With integrated circuits, many identical circuits can be fabricated on a single slice of silicon, with identical performance between circuits and at a low manufacturing cost. Moreover, the design and tooling costs for a new type of gate can be spread over many manufactured units.

In this description of binary addition, boxes have been labeled AND and OR; the overall function is controlled by how these boxes are interconnected. This is an important aspect of logic-circuit design. If the gates operate satisfactorily on binary 0, 1 basis, the design of the logic system can be completely carried out on paper. From the logic-system viewpoint, no matter what the boxes contain—relays, vacuum tubes, magnetic cores, transistors, or integrated circuits—the overall logic function will be the same. Which of these devices to use is determined by such factors as cost, size, power requirement, speed, and reliability.

(a)

	A_3	A_2	A_1	
	B_3	B_2	B_1	+
C	S_3	S_2	S_1	

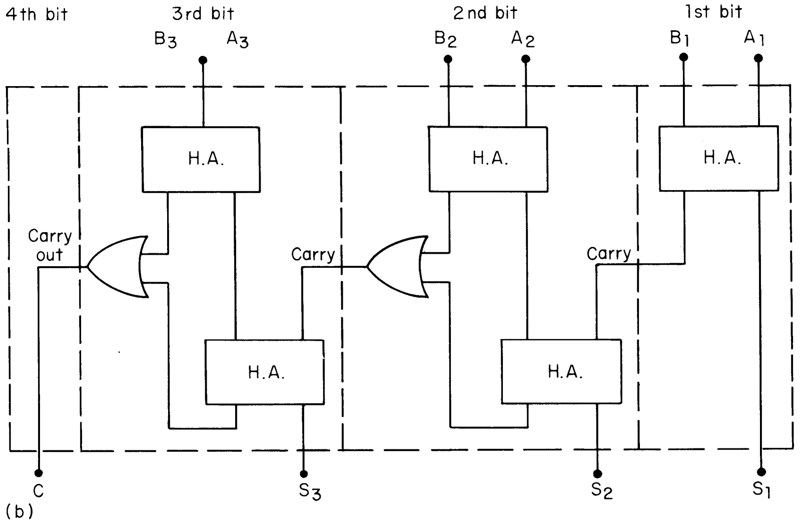


Fig. 1.8. The 3-bit parallel adder: (*a*) addition table; (*b*) arrangement.

1.4 LOGIC DESIGNATION

Logical 1 and 0 are represented, in most modern logic systems, by voltage levels. There are generally accepted rules for the definition of these logic levels in digital systems: *positive* logic (or active high levels) means that the most positive logic voltage level (also referred to as the high level) is defined to be the logical 1 state, and the most negative logic voltage level (also referred to as the low level) is defined to be the logical 0 state. *Negative* logic (or active low levels) is just the opposite—the most positive (high) level is a 0, and the most negative (low) level is a 1.

The effect of changing from one logic designation to the other is that all logic functions are complemented; for example, an AND becomes an OR, a NOR becomes a NAND, etc. The simplest approach to converting the logic designation (positive or negative) is to replace all 0s with 1s and all 1s with 0s in the truth table for the device, and then determine the resulting logic function. Chapter 6, "Combinational Logic Design," offers some detail on this subject.

The choice of positive or negative logic is made by the individual logic designer, largely as a matter of personal preference. There is no real advantage to either designation. Most logic designers and textbooks on logic design use positive logic, and positive logic is used throughout this book. However, in an attempt to make data on specific logic elements as general as possible, there is a trend toward specifying H (high level) or L (low level) in truth tables rather than 0s and 1s. Such a table is accurately called a *function table* rather than a *truth table.* But the latter term prevails, causing concern only to the purist.

The use of H and L in a function table eliminates the need to specify whether a truth table is stated in positive or negative logic, but it may temporarily confuse a person who normally thinks in terms of the strict Boolean terms of 0 and 1. The terms "positive" and "negative" logic have no place in pure logic theory. It is necessary to consider logic designation only when the physical implementation of a logic function is under consideration.

Accordingly, positive logic is understood in all discussions in this book unless the contrary is specified.

1.5 COMPARISON OF LOGIC TYPES

This brief comparison of logic types will explain the ascendancy of the transistor-transistor logic (TTL) family. Although the various logic types are identical as to functions performed, logic families can be distinguished by how they perform these functions.

Direct-coupled Transistor Logic (DCTL). A DCTL NOR gate is shown in Fig. 1.9. The input voltage is normally taken from the collector of the previous gate, and the output connects directly to the input of the following gate as indicated by the dashed-line circuits. If a logical 1 is input to *A* or *B* or *C,* the respective transistor saturates and the output voltage drops to its saturation voltage, giving a 0 output. With the dashed-line driving and load gates connected, the logic voltage swing at both input and output of the NOR gate will be from about 0.2 V for 0 to 0.9 V for 1. Threshold voltage is about 0.7 V. The advantage of the system is simplicity.

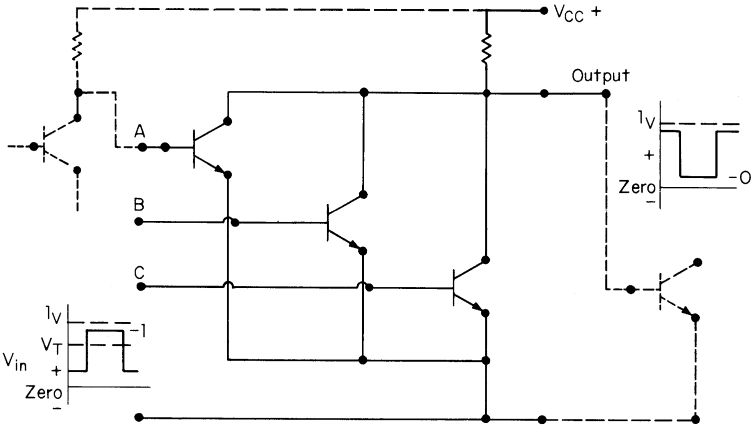


Fig. 1.9. Basic DCTL circuit.

The chief disadvantage of DCTL is that operation is affected by slight differences in characteristics of the individual transistors. If one transistor has a base-emitter voltage slightly lower than that of transistors in parallel with it, this transistor takes most of the available current and thus prevents proper overall operation of the circuit; this is called *current hogging.* To reduce the effect, resistors are included in series with each base lead so that the base current is less dependent on the individual base-emitter characteristics. The circuit then becomes resistor transistor logic (RTL).

Resistor Transistor Logic (RTL). This was the first family of logic circuits established as a standard catalog line. The basic arrangement in Fig. 1.10 shows the series resistors added to each transistor. Reducing the current-hogging effect with resistors permits a larger fan-out. However, resistors slow the switching speed of the circuit. Input capacitance must now be charged and discharged through additional resistance, increasing the circuit's time constant. Thus, with RTL, there

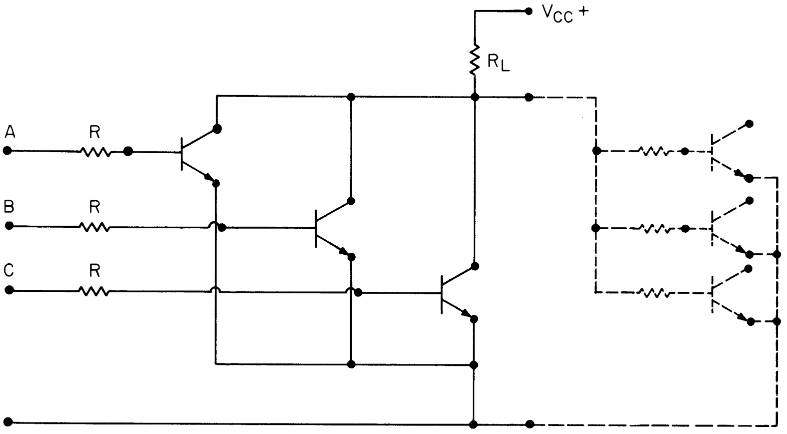


Fig. 1.10. Basic RTL circuit.

must be a compromise between fan-out and switching speed. Typical values of fan-out are 4 and 5, with a switching delay of 50 ns. The operating points and logic voltage swing are similar to those for DCTL. RTL has relatively poor noise immunity. The noise margin from 0 to threshold voltage is about 0.5 V, and only 0.2 V from 1 to threshold voltage.

Resistor-capacitor Transistor Logic (RCTL). RTL switching speed can be improved by adding a capacitor in parallel with the series resistor. This variation is called resistor-capacitor transistor logic (RCTL) and is shown in Fig. 1.11. The capacitor allows the leading and trailing edges of a signal pulse to bypass the resistor so that the transistor input capacitance charges more quickly. The use of the capacitor also allows higher values of resistance, making possible lower power dissipation per gate. The RCTL circuit is less than ideal from the viewpoint of fabrication because it includes a high proportion of resistors and capacitors. Capacitors and high-value resistors are relatively expensive in monolithic IC form because of the large area they require. RTL and RCTL circuits are still used in established equipment but seldom appear in new designs.

Diode Transistor Logic (DTL). A DTL logic circuit is shown in Fig. 1.12. Logic is performed by the input diode $D1$, $D2$, and $D3$. The signal is then coupled through a series diode D_S to an inverter stage consisting of a transistor and its load resistor. The overall DTL circuit is a NAND gate.

If all inputs are at a 1 with a positive signal voltage equal to V_{CC}, the three input diodes are reverse-biased and pass no current. The series diode D_S is forward-biased. Current flows through R_D and D_S into the base of the transistor and holds it in saturation. The low collector voltage $V_{CE(\text{sat})}$ gives a 0 at the output.

If any one input drops to ground potential, or goes low, the corresponding input diode conducts and current flows through R_D and that input diode. The potential at point X drops to the voltage across the input diode, about 0.7 V. This potential is not sufficient to drive current through D_S and the transistor base-emitter junction. Thus, the transistor is off and its collector potential rises to V_{CC}, giving a logical output.

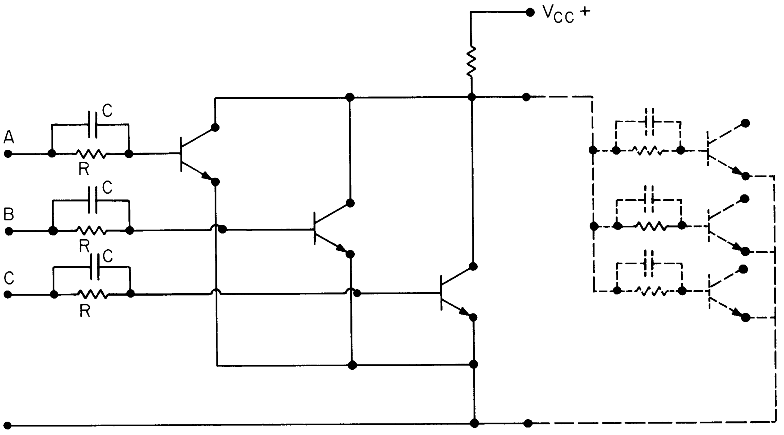


Fig. 1.11. Basic RCTL circuit.

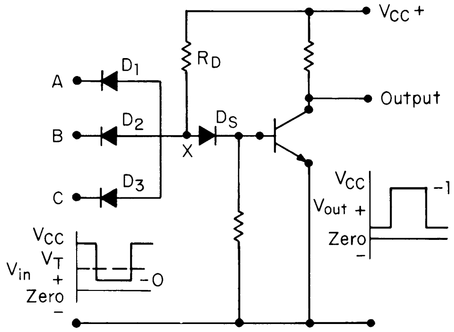


Fig. 1.12. Basic DTL circuit.

Under the former condition that all inputs are at a positive V_{CC} potential, the input diodes are reverse-biased and current flows through R_D and D_S into the base of the transistor. The potential at point X is approximately 1.4 to 0.7 V across the series diode D_S, and 0.7 V across the transistor base-emitter junction. Now, assume one input voltage is gradually reduced. Before the associated input diode can start to conduct, input voltage must be reduced to 0.7 V so that there is a forward voltage of 0.7 V across the diode. When the input diode conducts, point X voltage falls to 0.7 V and the transistor is off. Thus, the threshold voltage of this circuit is 0.7 V. If two diodes in series are used for D_S, threshold voltage is increased by an additional 0.7 V, to 1.4 V. This is usually the case in practical DTL circuits.

With a 1-input, the input resistance of the gate is high; the diodes are reverse-biased, and the gate does not load the previous circuit. Thus, outputs from a driving DTL circuit can be full supply voltage V_{CC}. If V_{CC} is 4 V, the two operating points at the input and output of the gate are 0.2 V for logical 0 and 4 V for logical 1. If two series diodes are used with a threshold voltage of 1.4 V, the logical-0-state noise margin is 1.2 V—substantially better than that of RCTL.

The DTL circuit switches faster than the RTL circuit, because the signal passes through the low forward resistance of the diodes to the transistor. A typical delay time is 25 ns. A fan-out as high as 8 is possible because of the high input impedance of the subsequent gates in the 1 state. The use of diodes rather than resistors and capacitors makes the DTL circuit more economical in IC form.

Transistor-transistor Logic (TTL). Operation of TTL circuits will be dealt with later in detail. It is important, however, to note major differences of TTL from other gate types. Figure 1.13 is the basic circuit of a TTL NAND gate. A single multiemitter transistor replaces input diodes and the series diode of DTL. Each emitter-base diode serves as one input, and the base-collector diode functions as the series diode. The multiemitter transistor is economically fabricated in monolithic form. A single isolated collector region is diffused, a single base region is diffused and formed in the collector region, and the several emitter regions are diffused as separate areas into the base region.

An output stage using an active pull-up transistor is added to the basic logic circuit to give current-gain drive for switching in both directions. This output configuration results in faster switching speed and higher fan-out capability.

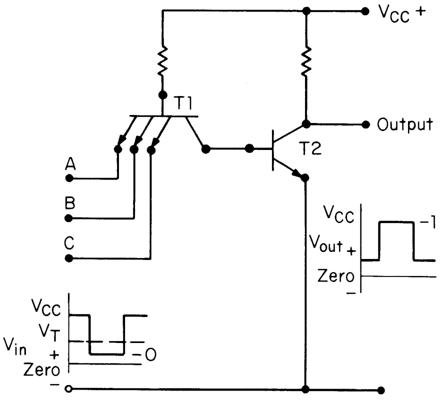


Fig. 1.13. Basic TTL circuit.

The TTL circuit is adaptable to virtually all forms of IC logic and produces the highest performance-to-cost ratio of all logic types.

Emitter-coupled Logic (ECL). A basic emitter-coupled logic gate is shown in Fig. 1.14. The emitters of logic transistors *T*1, *T*2, and *T*3 are coupled to the emitter of a reference transistor *T*4. The common-emitter resistor R_E is high enough in resistance to act as a constant-current source. The base of transistor *T*4 is connected to a reference voltage V_{bb}. If the inputs are all near ground potential (logical 0), *T*1, *T*2, and *T*3 are all off. No current flows through *R*1, and the common-collector potential rises toward V_{CC}. This drives *T*5 into conduction, and the output from the emitter of *T*5 goes positive to give a logical 1 output.

If one of the inputs is made positive and higher than the value of V_{bb} (logical 1), current flows through the associated transistor, causing the collector potential to fall and the output voltage from *T*5 also to fall, giving a logical 0 output. Since resistance R_E constitutes a constant-current source, as current through the logic transistor increases, current through the reference transistor decreases. The switching threshold voltage is equal to the reference voltage V_{bb}. Emitter coupling prevents

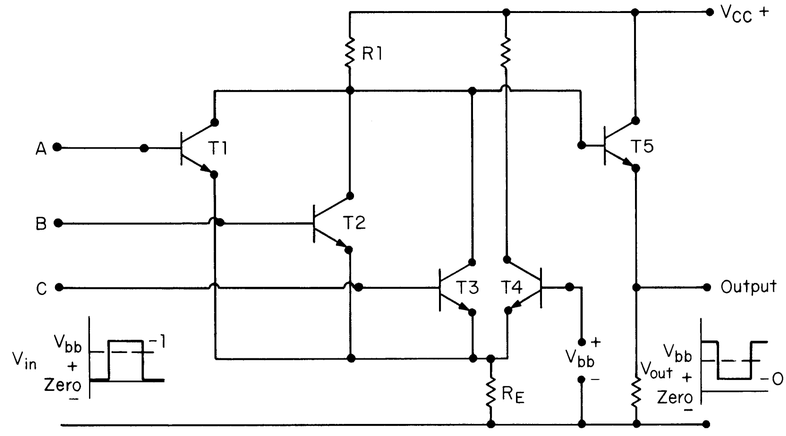


Fig. 1.14. Basic ECL circuit.

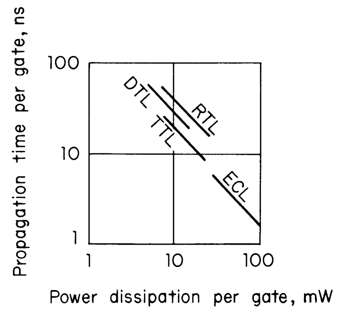


Fig. 1.15. Propagation time versus power spectrum of logic families.

the transistors from going into saturation. The result is very fast switching speed, typically only a few nanoseconds. Power dissipation, however, is relatively high—typically 50 mW. ECL is presently used in only the largest computers, where many disadvantages can be suffered for the sake of high speed. Since the input threshold voltage is equal to the reference voltage V_{bb}, it is well defined, but the 0 and 1 levels cannot be defined as precisely as in saturated circuits. The use of an emitter-follower output circuit gives a very low output impedance, allowing a very high fan-out of up to 25.

All the basic digital circuits described are available as standard types. In the detailed design of each family of circuits there is a compromise between switching speed and power dissipation, depending upon the specific application requirements. Figure 1.15 summarizes the relationship between typical propagation time and power dissipation for the various logic families.

1.6 LEVELS OF INTEGRATED-CIRCUIT COMPLEXITY

The first objective in the development of integrated circuits was the fabrication of a complete gate on a single silicon chip and its encapsulation in a suitable package. It soon became clear, however, that several similar gates can be fabricated on a single chip at very little additional cost. The next step, therefore, was to obtain the lowest cost per gate by forming as many gates as possible on one chip and encapsulating it in one package. This increasing complexity, coupled with the variety of logic-circuit types, has generated the need for definitions of complexity levels.

It is generally accepted that conventional digital integrated circuits contain 1 to 12 equivalent logic gates. At Texas Instruments this level of integration is known as small-scale integration (SSI). Medium-scale integration (MSI) refers to devices containing from 13 to 99 equivalent gates. Large-scale integration (LSI) covers the complexity range beyond 99 gates. These definitions refer to *monolithic* structures of semiconductor material and do not include hybrid assemblies.

The trend toward higher complexity levels has been motivated not only by the obvious advantages of reduced size and weight, but by the more significant advantage of the higher reliability that results from minimizing interconnections. There are other less obvious advantages as well. Figure 1.16 illustrates how MSI has affected one product. In 1963, the circuit on the left required 36 transistors and 244 diodes

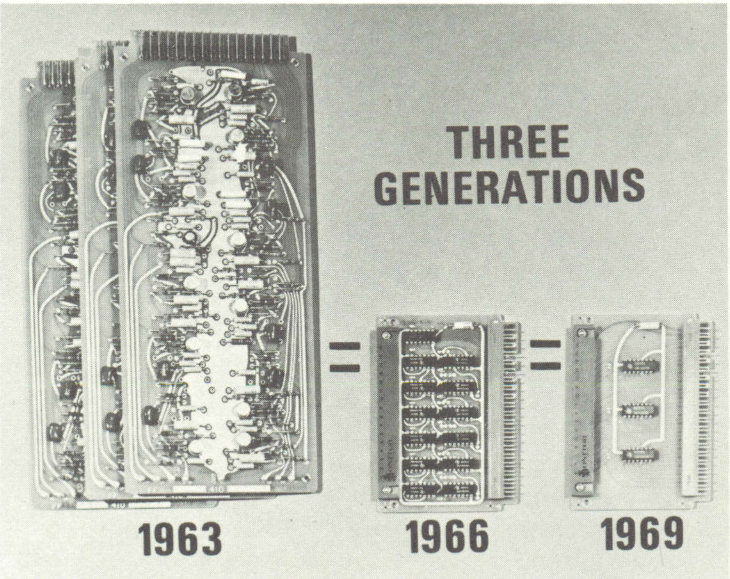


Fig. 1.16. Three generations of printed circuit boards.

to implement three 4-bit counters. By 1966, SSI performed the same function with 13 integrated circuits. In 1969, the development of MSI counter functions permitted the same capability in a single package, so that the board on the right is identical to the other two boards in function.

By comparison there occurred an 89 percent reduction in printed-circuit-board area. Also, two out of three sets of plug-in connectors and over half the cost of the larger enclosure for the system were saved. Even more important, system-design time was saved because the IC counters were already designed; production assembly labor time was reduced because only four components must be mounted as compared to 280 originally; systems check-out and test were simplified because 50 test points replaced 600; and the system builder enjoys reductions in inventory, spare-parts stocking, materials handling, vendor negotiations, and paper work.

2

Series 54/74 Overview

The Texas Instruments transistor-transistor logic family was introduced as a standard product line in 1964. Designated semiconductor network (or SN) Series 54 and implemented in the circuit configuration of Fig. 2.1, the original devices were intended primarily for the military market, where size, power consumption, and reliability requirements were paramount. Soon TI was able to offer these circuits as Series 74—lower-cost industrial versions with guaranteed operating characteristics over a 0 to 70°C range. Series 54 continues to cover the −55 to 125°C range.

At this writing, the Series 54/74 TTL family has grown and evolved into four major divisions: standard (SN54/74), high-speed (SN54H/74H), low-power (SN54L/74L), and Schottky-diode-clamped (SN54S/74S). Although the high-speed and low-power series were designed for specific applications, all four families are compatible and will interface directly with one another. This broad spectrum of speed/power combinations enables the designer to optimize all portions of a system according to his required-performance specifications.

2.1 TYPICAL CHARACTERISTICS

Not only are individual members of the Series 54/74 TTL families compatible, but they have the following typical characteristics in common:

Supply voltage	5.0 V
Logical 0 output voltage	0.2 V
Logical 1 output voltage	3.0 V
Noise immunity	1.0 V

When circuits of different series are combined, there are additional loading rules that should be observed. Loading guidelines are given in Chapter 4, "Extended-range Operation."

2.2 STANDARD SERIES 54/74 TTL

Standard Series 54/74 integrated circuits offer a combination of speed and power dissipation best suited to most applications. The basic standard gate circuit (Fig. 2.1) features a multiple-emitter input and an active pull-up output configuration. This multiple-emitter input transistor *Q*1 offers the most logic for the least physical size, and it is a major contributor to the fast switching speed of TTL. Low output impedance is attained with the active pull-up output transistor *Q*3, which also results in improved noise immunity and faster switching.

As shown in Table 2.1, circuits offered in the standard Series 54/74 line include shift registers, counters, decoders, memories, data selectors, and arithmetic elements in addition to the small-scale integration (SSI) devices. The circuits also include

Table 2.1. Standard TTL Integrated Circuits

Small-scale integration (SSI)	
NAND/NOR gates:	
Quad 2-input NAND	SN54/7400
Quad 2-input NAND, open collector	54/7401
Quad 2-input NOR	54/7402
Quad 2-input NAND, open collector	54/7403
Hex inverters	54/7404
Hex inverter buffer/driver, open collector	54/7406
Hex buffer/driver, open collector	54/7407
Quad 2-input AND	54/7408
Quad 2-input AND, open collector	54/7409
Triple 3-input NAND	54/7410
Hex inverter buffer/driver, open collector	54/7416
Hex buffer/driver, open collector	54/7417
Dual 4-input NAND	54/7420
Quad 2-input NAND, open collector	54/7426
8-input NAND	54/7430
Dual 4-input NAND buffer	54/7440
AND-OR-INVERT gates:	
Expandable dual 2-wide 2-input	SN54/7450
Dual 2-wide 2-input	54/7451
Expandable 4-wide 2-input	54/7453
4-wide 2-input	54/7454
Expanders:	
Dual 4-input	SN54/7460
Flip-flops:	
Edge-triggered *J-K*	SN54/7470
J-K master-slave	54/7472
Dual *J-K* master-slave	54/7473
Dual *D*-type edge-triggered	54/7474
Dual *J-K* master-slave, preset and clear	54/7476
J-K master-slave	54/74104
J-K master-slave	54/74105
Dual *J-K* master-slave	54/74107
Monostable nonretriggerable	54/74121

Table 2.1. Standard TTL Integrated Circuits (Continued)

Medium-scale integration (MSI)	
Decoders:	
BCD-to-decimal (4-to-10 lines)	SN54/7442
Excess-3-to-decimal (4-to-10 lines)	54/7443
Excess-3-Gray-to-decimal (4-to-10 lines)	54/7444
BCD-to-decimal decoder/driver (4-to-10 lines)	54/7445
BCD-to-7 segment decoder/driver	54/7446
BCD-to-7 segment decoder/driver	54/7447
BCD-to-7 segment decoder/driver	54/7448
BCD-to-7 segment decoder/driver	54/7449
BCD-to-decimal decoder/driver	SN74141
BCD-to-decimal decoder/driver	SN54/74145
4-line-to-16-line decoder/demultiplexer	54/74154
Dual 2-line-to-4 line decoder/demultiplexer	54/74155
Dual 2-line-to-4 line decoder/demultiplexer, open collector	54/74156
Memories/latches:	
4-bit bistable latches	SN54/7475
4-bit bistable latches	54/7477
16-bit active element memory	54/7481
16-bit active element memory	54/7484
256-bit read-only memory	SN7488
8-bit bistable latches	SN54/74100
Arithmetic elements:	
Gated full-adder	SN54/7480
2-bit full-adder	54/7482
4-bit full-adder	54/7483
Quad 2-input exclusive-OR gate	54/7486
4-bit arithmetic logic unit	54/74181
Look-ahead carry generator	54/74182
Counters:	
Decade	SN54/7490
Divide-by-12	54/7492
4-bit binary	54/7493
Synchronous decade up/down	54/74192
Synchronous 4-bit up/down	54/74193
Shift registers:	
8-bit	SN54/7491A
4-bit	54/7494
4-bit	54/7495
5-bit	54/7496
Data selectors/multiplexers:	
16-bit	SN54/74150
8-bit, with strobe	54/74151
8-bit	54/74152
Dual 4-line-to-1-line	54/74153
Miscellaneous:	
8-bit odd/even parity generator/checker	SN54/74180

a wide choice of flip-flops: single and dual, edge-triggered or master-slave, *D*-type or *J-K* input. A versatile one-shot and a wide variety of gate circuits, including open-collector output gates, are also available.

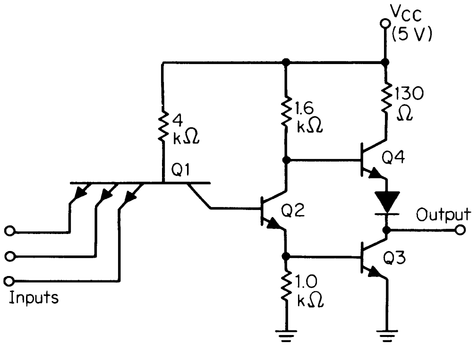


Fig. 2.1. Original SN54/74 TTL gate.

2.3 SERIES 54/74 LOW-POWER TTL

Low-power circuits, designated Series 54L/74L, give the best speed-power product of all logics available. The basic low-power gate circuit as shown in Fig. 2.2 has essentially the same configuration as that of the standard Series 54/74 gate; only the resistor values are increased. Since an increase in resistance results in a reduction of power dissipation, the power requirements of low-power gates are less than one-tenth of those of standard ICs. Series 54L/74L devices have a power dissipation of only 1 mW per gate and speeds approximately twice as fast as other circuits with similar power dissipation. A speed of 33 ns per gate is typical.

Series 54L/74L is ideal for applications where power consumption and heat dissipation are the critical parameters. Table 2.2 lists the low-power SSI and MSI circuits now available.

2.4 SERIES 54H/74H HIGH-SPEED TTL

The circuit configuration of the high-speed gate (Fig. 2.3) is basically the same as that of the standard Series 54/74 gate. However, resistor values are lower and clamping diodes are included on each multiple-emitter input to reduce transmission-line effects that become more apparent with the faster rise and fall times.

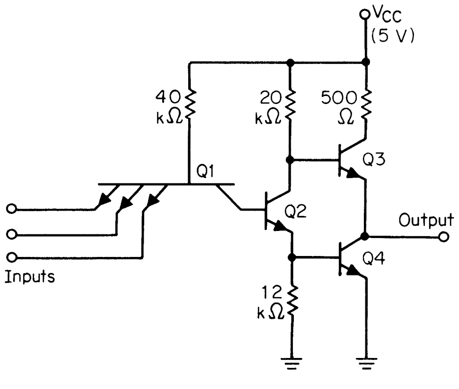


Fig. 2.2. Low-power TTL gate SN54L/74L.

Table 2.2. Low-power TTL Integrated Circuits

Small-scale integration (SSI)	
NAND/NOR gates:	
Quad 2-input NAND	SN54/74L00
Hex inverter	54/74L04
Triple 3-input NAND	54/74L10
Dual 4-input NAND	54/74L20
8-input NAND	54/74L30
AND-OR-INVERT gates:	
Dual 2-wide	SN54/74L51
4-wide 3-2-2-3-input	54/74L54
2-wide 4-input	54/74L55
Flip-flops:	
R-S master-slave	SN54/74L71
J-K master-slave	54/74L72
Dual *J-K* master-slave	54/74L73
Dual *D*-type edge-triggered	54/74L74
Dual *J-K* master-slave	54/74L78
Medium-scale integration (MSI)	
Arithmetic elements:	
Quad 2-input exclusive-OR	SN54/74L86
Counters:	
4-bit binary	SN54/74L93
Shift registers:	
8-bit	SN54/74L91
4-bit right/left shift	54/74L95
4-bit data selector/storage register	54/74L98
4-bit right/left shift	54/74L99
Miscellaneous:	
4-bit magnitude comparator	SN54/74L85

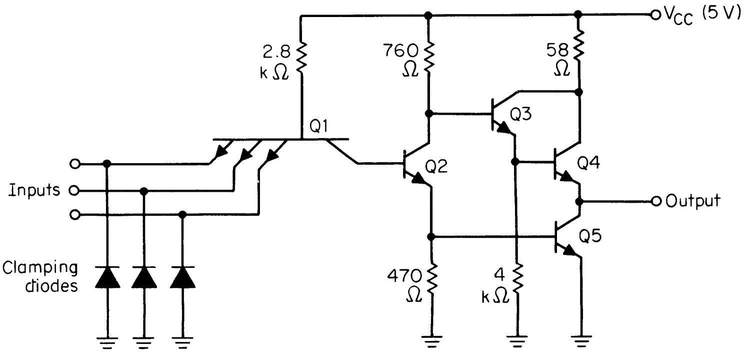


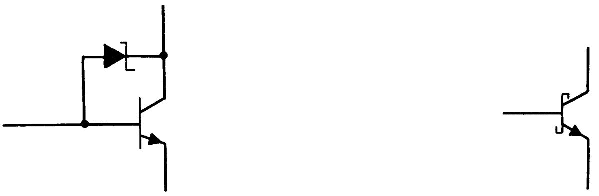
Fig. 2.3. High-speed TTL gates, SN54H/74H.

The output section consists of a Darlington transistor pair $Q3$ and $Q4$. This arrangement provides slightly higher speed (6 ns per gate) than the standard gate because of transistor action and low steady-state output impedance, which is typically 10 Ω in the unsaturated state and 100 Ω in the saturated state. However, high-speed TTL circuits, Series 54H/74H, have the disadvantage of using more power than the standard Series 54/74 gate.

Table 2.3 lists the available SSI and MSI circuit functions, including both AND

Table 2.3. High-speed TTL Integrated Circuits

Small-scale integration (SSI)	
NAND/NOR gates:	
Quad 2-input NAND	SN54H/74H00
Quad 2-input NAND, open collector	54H/74H01
Hex inverter	54H/74H04
Hex inverter, open collector	54H/74H05
Triple 3-input NAND	54H/74H10
Triple 3-input AND	54H/74H11
Dual 4-input NAND	54H/74H20
Dual 4-input AND	54H/74H21
Dual 4-input NAND, open collector	54H/74H22
8-input NAND	54H/74H30
Dual 4-input NAND buffer	54H/74H40
AND-OR gates and AND-OR-INVERT gates:	
Expandable dual 2-wide 2-input AND-OR-INVERT	SN54H/74H50
Dual 2-wide 2-input AND-OR-INVERT	54H/74H51
Expandable 4-wide 2-2-2-3-input AND-OR	54H/74H52
Expandable 4-wide 2-2-2-3-input AND-OR-INVERT	54H/74H53
2-2-2-3-input AND-OR-INVERT	54H/74H54
Expandable 2-wide 4-input AND-OR-INVERT	54H/74H55
Expanders:	
Dual 4-input	SN54H/74H60
Triple 3-input	54H/74H61
3-2-2-3-input AND-OR	54H/74H62
Flip-flops:	
J-K master-slave	SN54H/74H71
J-K master-slave	54H/74H72
Dual *J-K* master-slave	54H/74H73
Dual *D*-type edge-triggered	54H/74H74
Dual *J-K* master-slave with preset and clear	54H/74H76
Dual *J-K* master-slave	54H/74H78
J-K, negative edge-triggered	54H/74H101
J-K, negative edge-triggered	54H/74H102
Dual *J-K*, negative edge-triggered	54H/74H103
Dual *J-K*, negative edge-triggered	54H/74H106
Dual *J-K*, negative edge-triggered	54H/74H108
Medium-scale integration (MSI)	
Arithmetic elements:	
4-bit true/complement, zero/one element	SN54H/74H87
Dual carry-save full-adder	54H/74H183



(a) Transistor and Schottky barrier diode clamp

(b) Symbol for transistor with Schottky barrier diode clamp

Fig. 2.4. Schottky-clamped TTL, SN54/74S.

and NAND gates, single-wire expanders, and open-collector output gates to facilitate the wire-AND function.

Series 54H/74H also includes master-slave flip-flops and edge-triggered flip-flops capable of operation with clock-input frequencies as high as 50 MHz.

2.5 SERIES 54S/74S SCHOTTKY-CLAMPED TTL

Series 54S/74S has the highest speed in the 54/74 line. It combines the high speed of unsaturated logic (ECL) with the relatively low power consumption of TTL. This performance is achieved by using a Schottky-barrier diode (SBD) as a clamp from base to collector as shown in Fig. 2.4. The features of the SBD that make it useful are that it is free from minority carriers and therefore has no stored charge, and that the SBD has a lower forward voltage drop than a silicon p-n junction. When used as a clamp, the SBD diverts most of the excess base current and prevents the transistor from reaching "classic" saturation. With virtually no stored charge in either the SBD or the transistor, there is a great reduction in storage time, which significantly improves transistor switching time.

The output has been modified on the 54S/74S types to give a symmetrical transfer characteristic (see Fig. 2.5). *Q*3 and the associated resistors replace the resistor-to-ground used in the other types of 54/74 series.

Outstanding features of Series 54S/74S are 3-ns average propagation delay time and an average power dissipation per gate of 20 mW.

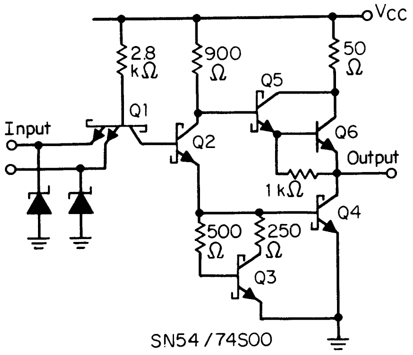


Fig. 2.5. Schottky-barrier diode clamp.

3

Circuit Analysis and Characteristics of Series 54/74

3.1 BASIC TTL OPERATION

The basic NAND gate is shown in Fig. 3.1; *all* inputs must be at logical 1 in order that the output be at a logical 0 (*any* input at logical 0 will give a logical 1 out). The NAND function itself is accomplished from the input of $Q1$ to the collector of $Q2$, and transistors $Q3$ and $Q4$ make up the output drive circuitry.

An understanding of circuit operation can be facilitated by reference to the voltage transfer characteristic, Fig. 3.2. If V_I at one or more of the input emitters is less than V_a, transistors $Q2$ and $Q3$ are turned off. This causes transistor $Q4$ to conduct, resulting in a logical 1 output. The logical 1 output can be determined approximately from

$$V_{OH} = V_{CC} - V_{BE(Q4)} - V_F$$

where V_F is the forward voltage drop of the diode. With the input voltage less than V_a, current flows out of the input emitter of $Q1$ and is determined primarily by V_{CC} and $R1$. $Q4$ acts as an emitter-follower, providing a low-impedance driving source in the logical 1 state.

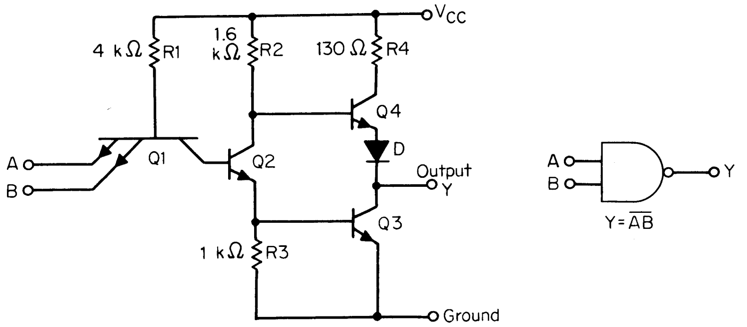


Fig. 3.1. Schematic diagram and logic symbol for SN54/74 TTL NAND gate.

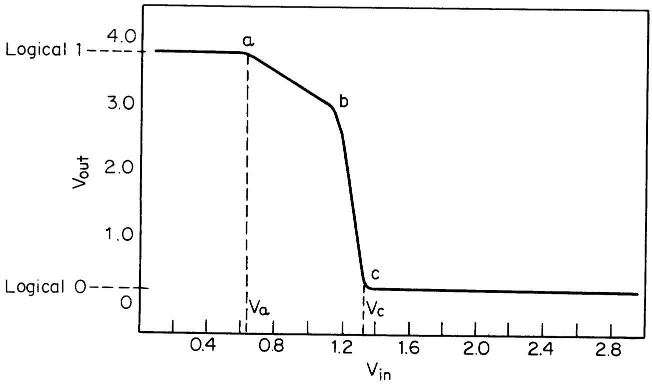


Fig. 3.2. TTL voltage transfer characteristics.

When V_I of all inputs is greater than V_c (Fig. 3.2), transistors *Q*2 and *Q*3 conduct; *Q*4 is off, and the output is at a logical 0. The logical 0 level is determined by the saturation resistance of *Q*3 and the current it must sink. *Q*1 is now operating in its inverse active region and requires an input current of $I_B h_{FE(\text{inv})}$. Where h_{FE} is a forced beta condition, its value is much less than 1.

For purposes of discussion, the input levels defined by $V_a < V_I < V_c$ will be considered the transition region where translation occurs from a logical 1 output to a logical 0 output. Assume that all inputs are tied together, and V_I is increased from 0. *Q*1 base current is gradually diverted from the emitter of *Q*1 to the collector of *Q*1, causing *Q*2 to conduct. This conduction occurs at $V_I \approx 0.7$ V (point *a* on Fig. 3.2). *Q*2 is now operating in its linear region with a gain from input to collector determined by the ratio *R*2/*R*3. Since *Q*4 remains on, the output follows the gain characteristic of *Q*2 and therefore decreases at a slope of 1.6 (point *a* to *b* on transfer curve). At point *b* the input is high enough to cause *Q*3 to conduct. This conduction effectively reduces the emitter impedance of *Q*2 and increases its gain; hence the steep slope region between points *b* and *c*. *Q*4 turns off at point *c*, and the output is at the logical 0 state.

Power-supply current is different from one output state to another, because the number of "on" devices changes. During the switching transition there is a period when *Q*2, 3, and 4 are all on. This results in I_{CC} current spiking, which is discussed in Chapter 5, "Noise Considerations."

3.2 TTL ADVANTAGES

As shown in Fig. 3.1, the multiple-emitter input transistor *Q*1 provides some advantages not otherwise attainable. When the voltage at any input is low (logical 0), that base-emitter junction is forward-biased. For this condition, conventional n-p-n transistor action requires a large current flow into the collector terminal. Since the collector circuit of *Q*1 is also the base circuit of *Q*2, and since large reverse base current is not possible, *Q*1 saturates. The forward-biased collector-base junction provides an extremely low impedance path for fast removal of $Q2$ stored base charge. Turn-off time is therefore much better than in other gate configurations.

The multiple-emitter transistor replaces combinations of diodes, resistors, and transistors found in other logic types. By comparison then, its geometrical size is small. Smaller size yields lower costs or more functions, or both, per given IC chip. Smaller size also reduces parasitic capacitance, and faster switching speed results.

On the output side of the TTL gate, $Q3$ and $Q4$ make up what is referred to as a *totem-pole* or *active pull-up* output. The purpose is to provide a low driving source impedance. This arrangement permits a high-capacitance load to be driven without seriously degrading switching time. With the output in the logical 1 state, $Q4$ appears as an emitter-follower source driving current into the load. For a logical 0 output, current from the load is impeded only by the low saturation resistance of $Q3$.

3.3 CIRCUIT PARAMETERS

Specification sheets can be misleading unless test conditions are clearly stated. TI Series 54/74 TTL ICs are guaranteed at worst-case test conditions. A TI specification that a 54/74 TTL device has a fan-out of 10 can be taken at face value. With ten 54/74 TTL gate inputs connected to an output, the device is guaranteed to operate within specifications over its entire temperature range and supply voltage range. It is *not* necessary to limit the fan-out to 8 or 9 to assure that the device will always operate. The worst-case testing provides a built-in margin of safety. All d-c limits shown on the data sheet are guaranteed over the entire temperature range (−55 to +125°C for Series 54 and 0 to 70°C for Series 74) and the entire supply voltage range (4.5 to 5.5 V for Series 54, and 4.75 to 5.25 V for Series 74). A single minimum or maximum value is guaranteed over a temperature and supply voltage range, since the designer is limited by whatever happens to be the worst value of a particular parameter, regardless of the temperature and supply voltage at which it occurs. This practice gives added assurance of reliable system operation.

The following sections describe the terminal characteristics for a typical 54/74 TTL gate. They describe also the worst-case testing procedures used for the data sheet specifications. The curves represent the characteristics of a typical device at the time of writing. Although most devices have characteristics close to the average values shown, some devices may vary within the data sheet limits.

Transfer Characteristics. Since the most common application for a logic gate is to drive another similar logic gate, the input and output logic levels must be compatible. The input and output logic levels for Series 54/74 logic functions are defined as follows (symbols given first have superseded the symbols in parentheses, but the older forms are still seen; definitions are in positive logic):

V_{IL} (previously $V_{in(0)}$) is the voltage level required for a logical 0 at an input. It is a guaranteed maximum of 0.8 V.

V_{IH} (previously $V_{in(1)}$) is the voltage level required for a logical 1 at an input. It is a guaranteed minimum of 2 V.

V_{OL} (previously $V_{out(0)}$) is the voltage level output from an output in the logical 0 state. It is a guaranteed maximum of 0.4 V.

Table 3.1. Typical Output and Threshold Voltages for TI SN5400/7400 Gates

As function of ambient temperature T_A· $V_{CC} = 5$ V, $N = 10$

	−55°C	0°C	25°C	70°C	125°C
V_{OH}	3.0	3.1	3.25	3.3	3.5
V_{OL}	0.25	0.29	0.30	0.31	0.32
V_T	1.5	1.4	1.3	1.2	1.0

As function of supply voltage V_{CC}· $T_A = 25$°C, $N = 10$

	4.5 V	4.75 V	5.0 V	5.25 V	5.5 V
V_{OH}	2.6	2.85	3.25	3.35	3.55
V_{OL}	0.33	0.32	0.30	0.30	0.30
V_T	1.28	1.29	1.3	1.32	1.35

V_{OH} (previously $V_{out(1)}$) is the voltage level output from an output in the logical 1 state. It is a guaranteed minimum of 2.4 V.

V_T is the threshold voltage at which the input and output voltages are equal.

Typical values for these parameters are presented in Table 3.1 for TI Series 54/74 gates. These are *not* worst-case definitions, but typical operating values for the guaranteed fan-out of 10 ($N = 10$). Examination of Table 3.1 reveals that these voltage parameters are a function of both ambient temperature and supply voltage. Typical V_O *vs.* V_I transfer characteristics as a function of ambient temperature are shown in Fig. 3.3.

To assure the user of TI SN54/74 TTL circuits a safe operating margin, the devices are tested and guaranteed to the data sheet limits shown in Table 3.2.

These voltage values are guaranteed for a fan-out of 10 over the entire recommended supply voltage range and ambient temperature range.

Inspection of Table 3.2 reveals that the guaranteed maximum logical 0 output voltage ($V_{OL} \leq 0.4$ V) is 400 mV less than the guaranteed minimum logical 0 input voltage ($V_{IL} \leq 0.8$ V) required for a gate to operate correctly. Also, the guaranteed minimum logical 1 output voltage ($V_{OH} \geq 2.4$ V) is 400 mV greater than the guaranteed maximum logical 1 input voltage ($V_{IH} \geq 2.0$ V) required for a gate to operate

Table 3.2. Guaranteed Input/Output Parameters for TI SN5400/7400 Gates

	Ambient temperature range	Supply voltage range	Fan-out
SN5400	$-55°\text{C} \leq T_A \leq 125°\text{C}$	$4.5\text{ V} \leq V_{CC} \leq 5.5\text{ V}$	$N = 10$
SN7400	$0°\text{C} \leq T_A \leq 70°\text{C}$	$4.75\text{ V} \leq V_{CC} \leq 5.25\text{ V}$	
Guaranteed output parameters. . .		$V_{OH} \geq 2.4$ V	$V_{OL} \leq 0.4$ V
Guaranteed input parameters . . .		$V_{IH} \geq 2.0$ V	$V_{IL} \leq 0.8$ V

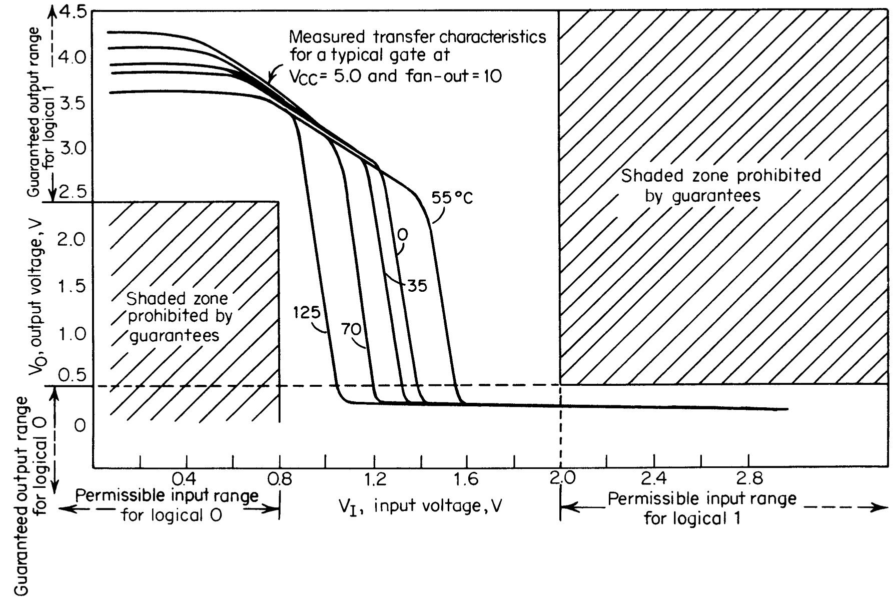


Fig. 3.3. Voltage transfer characteristics for typical SN54/74 TTL NAND gate.

correctly. This 400-mV margin of safety in both logic states is called the *guaranteed d-c noise margin;* it will be discussed further in Chapter 5, "Noise Considerations." Thus a 54/74 gate output is compatible with a 54/74 gate input and is capable of driving up to 10 of these gate inputs with a guaranteed 400-mV margin of safety.

Input Characteristics. Knowledge of the input and output characteristics of a 54/74 gate is necessary to utilize these devices fully. This knowledge is particularly necessary when a device interfaces with something other than 54/74 TTL elements or is used outside the guaranteed specifications (see Chapter 4, "Extended-range Operation"). For example, the expression "fan-out of 10" has no meaning when anything but other 54/74 gates are being driven. Knowledge of the voltage-current relationships for all elements involved is necessary for proper design.

Figures 3.4 and 3.5 illustrate typical input current (I_I) versus input voltage (V_I) characteristics for a 54/74 TTL gate input during normal operation. Any device used to drive a TTL gate must both source and sink current. Conventionally, current flowing toward a device terminal is designated positive, and current flowing away is negative. To comply with this convention, current-direction arrows are shown directed toward a device terminal, since this is the assumed positive direction. Thus, I_{IL} (previously $I_{in(0)}$) is a negative current since it flows out of the input terminal. Since I_{IH} (previously $I_{in(1)}$) flows into the input terminal, it is positive.

Figure 3.6 depicts a standard Series 54/74 input circuit showing the path for I_{IL}. During a logical 0 input state ($V_{IL} \leq 0.8$ V), the input current I_{IL} is primarily determined by resistor $R1$. However, I_{IL} is also a function of the supply voltage V_{CC}, the ambient temperature T_A, and the input voltage V_{IL} (see Figs. 3.4 and 3.5).

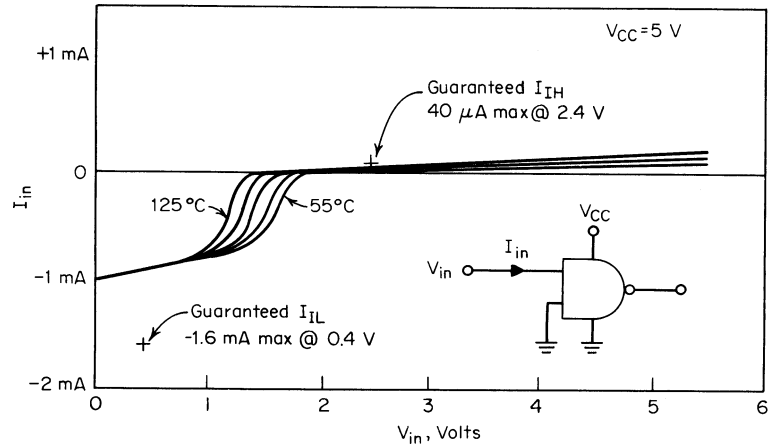


Fig. 3.4. Input characteristics as a function of temperature for standard Series SN54/74.

To assure desired device operation under all possible conditions, the worst-case test shown in Fig. 3.7 is performed on all devices. Note that V_{CC} is taken to its highest allowable value to maximize I_{IL}. With the exception of the input under test, all *unused* inputs are taken to a logical 1 to maximize any contribution of these inputs to I_{IL} of the emitter under test. This logical 1 test value (4.5 V) was chosen by noting that in the normal 54/74 totem-pole output circuit, there exists minimally one V_{BE} drop and a diode V_F drop (See Fig. 3.1) between supply and output. Therefore, I_{IL} is measured under the most generally prevailing worst-case conditions. The real concern is to screen out those devices with an I_{IL} of sufficient magnitude to cause greater than 0.4 V to be present in a gate output driving 10

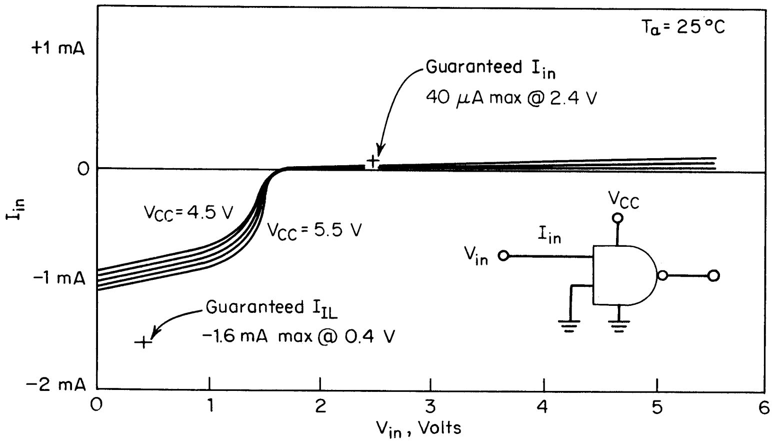


Fig. 3.5. Input characteristics as a function of supply voltage for standard Series SN54/74.

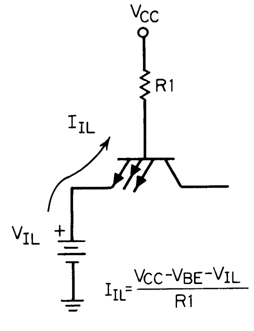
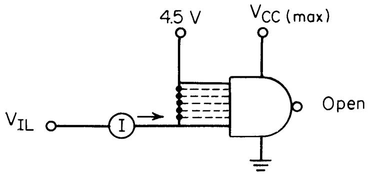


Fig. 3.6. Series SN54/74 input circuit illustrating I_{IL}.



1. $V_{IL} = V_{OL\ max} = 0.4$ V
2. $V_{CC} = 5.5$ V for series 54
3. $V_{CC} = 5.25$ V for series 74
4. Each input is tested separately

Fig. 3.7. Worst-case testing for I_{IL}.

such inputs ($V_{OL} \leq 0.4$ V at I_{OL} [previously $I_{out(0)}$] = 16 mA). Thus, all acceptable 54/74 gates have $I_{IL} \leq |-1.6|$ mA at $V_I = 0.4$ V; this current is a logical 0 load of $N = 1$ for a gate output.

Another input parameter that must be measured and controlled is input current in the logical 1 state I_{IH}. When all inputs of a multiple-emitter 54/74 TTL gate are at logical 1, the emitter-base junction is reverse-biased, and the collector-base junction is forward-biased. Thus, an emitter acts like a collector, and a collector acts like an emitter. Input current now is a function of the inverse current gain $h_{FE(\mathrm{inv})}$ of the input transistor $Q1$ (see Fig. 3.8). However, if one or more of the input emitters is a logical 0, inverse transistor action between emitters, or emitter-to-emitter leakage, may occur in addition to or in place of the inverse transistor

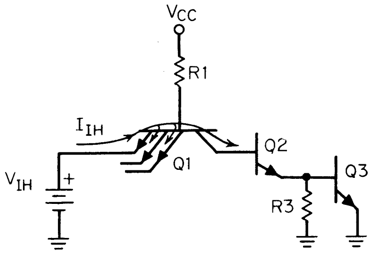


Fig. 3.8. Series SN54/74 input circuit illustrating I_{IH}.

action between the logical 1 emitter and the collector. Thus, I_{IH} current may split into several currents inside the TTL gate (see Fig. 3.8). Figures 3.4 and 3.5 show that I_{IH} is a function of supply voltage, ambient temperature, and actual input voltage V_{IH}.

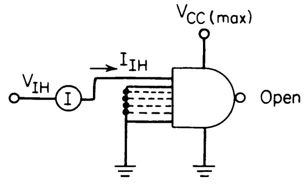
To assure desired device operation under all possible conditions, the worst-case test shown in Fig. 3.9 is performed on all devices. Note that all unused inputs are grounded, and V_{CC} is maximized to provide maximum I_{IH} current. This arrangement maximizes base current drive for inverse h_{FE} operation. The real concern is to screen out those devices with an I_{IH} of sufficient magnitude to cause a V_{OH} of less than 2.4 V to be present at the gate output driving 10 such inputs. An $I_{IH} \leq 40$ μA measured with V_I at 2.4 V is acceptable.

An additional I_{IH} test is performed to check for input emitter breakdown voltage. Again, the circuit of Fig. 3.9 is used, but a $V_{IH} = 5.5$ V. An acceptable device has an $I_{IH} \leq 1$ mA at 5.5 V. The breakdown voltage test at 5.5 V is performed to assure that an input emitter-base junction will not be damaged if a voltage source up to 5.5 V (maximum recommended V_{CC} for Series 54/74 devices) is applied to a gate input. The emitter-base junctions of a Series 54/74 TTL gate are physically very small, and they may be damaged if too much power is dissipated in the junction. Up to 5.5 mW ($I_{IH} \leq 1$ mA at 5.5 V) may safely be dissipated at a gate input without damaging the device. Further information is presented in Chapter 4, "Extended-range Operation."

From this discussion, it should be apparent that each input of a multiple-emitter TTL gate represents a load of 1 ($N = 1$) in both logical states. A logical 0 input load is $I_{IL} \leq |-1.6|$ mA at $V_{in} = 0.4$ V. A logical 1 input load is $I_{IH} \leq 40$ μA at $V_{in} = 2.4$ V.

Output Characteristics. The Series 54/74 gate output stage is designed for a fan-out of ten 54/74 gate inputs. An examination of the output test conditions will explain this more fully.

Figure 3.10 depicts that section of the output drive circuit which produces a logical 0 output voltage V_{OL}. Figures 3.11 and 3.12 illustrate the typical characteristics for



1. $V_{IH} = V_{OH\,min} = 2.4$ V
2. $V_{CC} = 5.5$ V for series 54
3. $V_{CC} = 5.25$ V for series 74
4. Each input tested separately

Fig. 3.9. Worst-case testing for I_{IL} at $V_{IH} = V_{OH}$ min.

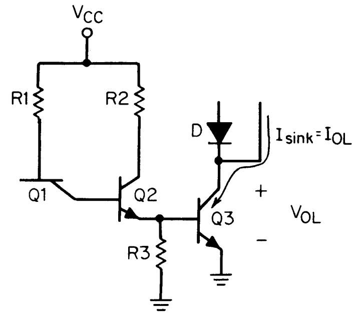


Fig. 3.10. Series SN54/74 output drive circuit illustrating V_{OL}.

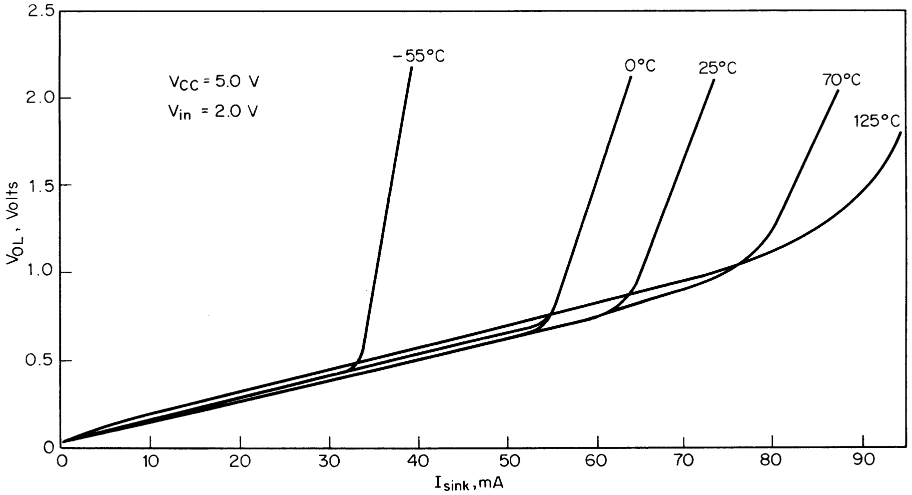


Figure 3.11

V_{OL} as a function of I_{OL} (sink current). Note that V_{OL} for constant I_{OL} decreases for increasing ambient temperature (Fig. 3.11) and increasing supply voltage (Fig. 3.12). Thus the output transistor ($Q3$ of Fig. 3.11) is more difficult to keep in saturation at lower temperatures and lower supply voltages. Accordingly, the minimum supply voltage is used for worst-case testing of V_{OL}. Figure 3.13 shows

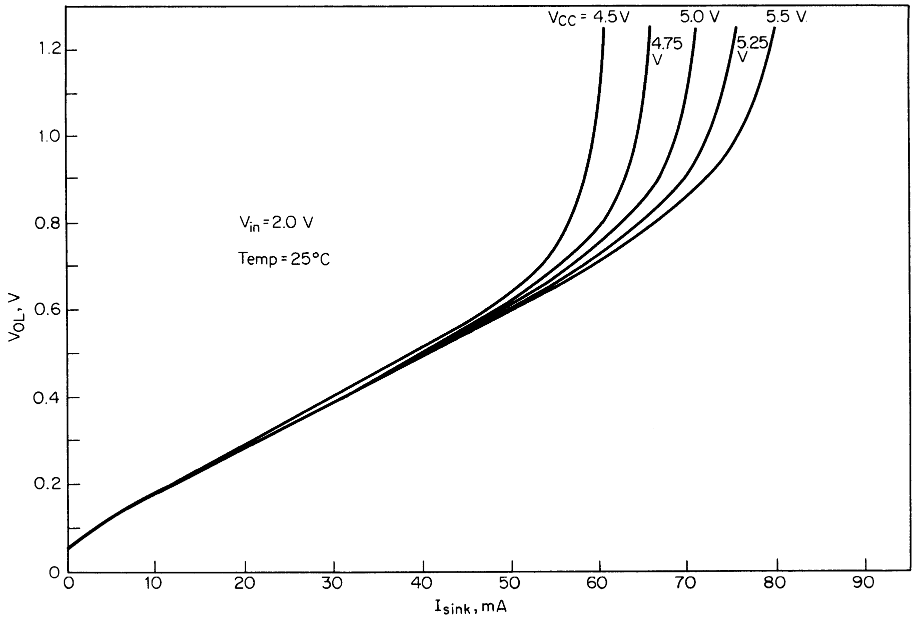
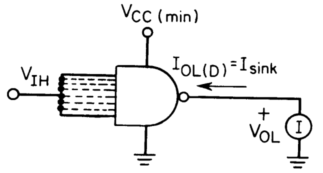


Figure 3.12



1. $V_{IH} = V_{IH\ min} = 2.0$ V
2. $I_{sink} = 16$ mA
3. $V_{CC} = 4.5$ for series 54
4. $V_{CC} = 4.75$ V for series 74
5. All inputs are tested simultaneously

Fig. 3.13. Worst-case testing for V_{OL}.

the method used for worst-case testing for V_{OL}. All inputs are connected to the minimum logical 1 input voltage (2.0 V). Since the gate was designed to drive ten 54/74 input logical 0 loads, it must sink 16 mA at $V_{OL} \leq 0.4$ V ($I_{IL} \leq |-1.6|$ mA at $V_{OL} = 0.4$ V) to pass this V_{OL} test.

Figure 3.14 shows that the logical 1 output drive circuit consists primarily of the emitter-follower transistor *Q*4 and resistors *R*2 and *R*4. Typical V_{OH} *vs.* I_{OH} (previously $I_{out(1)}$) characteristics are shown in Figs. 3.15 and 3.16. For constant I_{OH}, V_{OH} decreases with decreasing supply voltage and with decreasing temperature. In fact, the logical 1 output voltage follows changes in the supply voltage practically volt for volt.

When measuring worst-case logical 1 output levels, the minimum supply voltage is used (see Fig. 3.17). When a worst-case logical 0 input voltage (0.8 V) is applied to an input, and when that voltage is required to hold the output transistor *Q*3 (see Fig. 3.14) in the off state, unused inputs are tied to the supply voltage. This situation represents worst-case conditions, since this high voltage would tend to turn on the transistor *Q*3 if it were not for the logical 0 input. Since the gate was designed

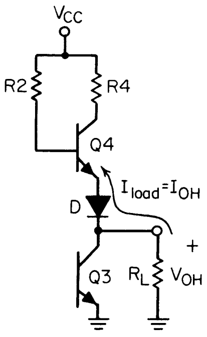


Fig. 3.14. Series SN54/74 output drive circuit illustrating V_{OH}.

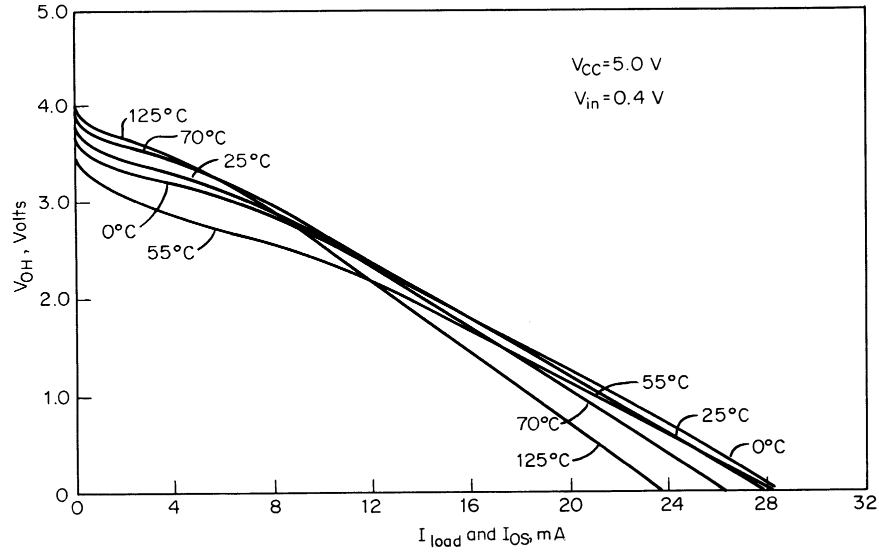


Figure 3.15

to drive ten 54/74 input logical 1 loads, it must source -400 μA at $V_{OH} \geq 2.4$ V ($I_{IH} \leq 40$ μA at $V_{in} = 2.4$ V) to pass this V_{OH} test.

A second I_{OH} test has been provided to determine the logical 1 output current when the output is shorted to ground; this is the I_{OS} test. This test checks the value of current-limiting resistor $R4$ and proper operation of transistor $Q4$ and diode D (see Fig. 3.14). Both a minimum and a maximum I_{OS} (short-circuit output current)

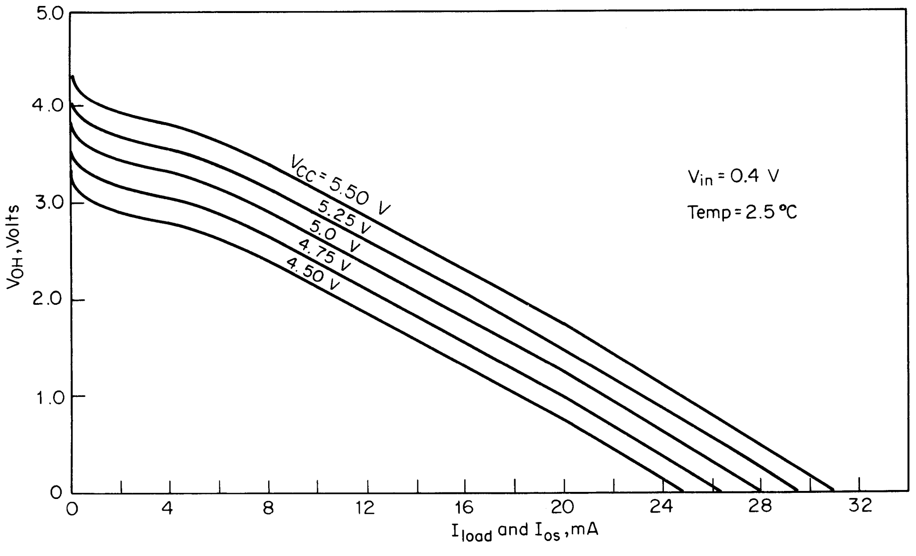
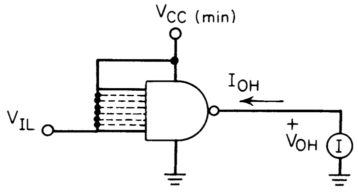
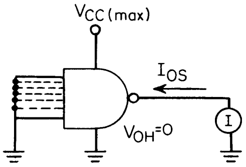


Figure 3.16


V_CC (min)
I_OH
V_IL
+
V_OH
I

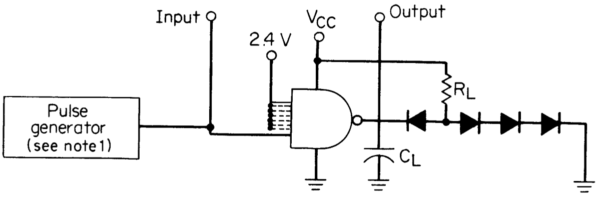
1. $V_{IL} = V_{IL\ max} = 0.8$ V
2. $I_{load} = -400\ \mu A$
3. $V_{CC} = 4.5$ V for series 54
4. $V_{CC} = 4.75$ V for series 74
5. Each input is tested separately

Fig. 3.17. Worst-case testing for V_{OH}.

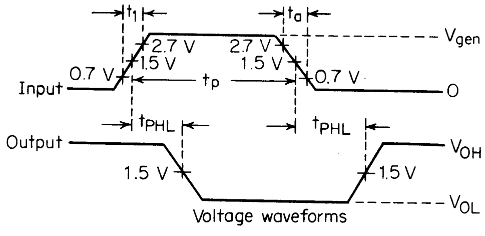

V_CC (max)
I_OS
V_OH=0
I

1. $V_{OH} = 0$
2. $V_{CC} = 5.5$ V for series 54
3. $V_{CC} = 5.25$ V for series 74
4. Each gate is tested separately

Fig. 3.18. Worst-case testing for I_{OS}.


Input
2.4 V
V_CC
Output
Pulse generator (see note 1)
R_L
C_L

Test circuit



Voltage waveforms

1. The generator has the following characteristic:
   $V_{gen} = 3.5$ V, $t_0 = 5$ ns, $t_1 = 10$ ns, $t_p = 0.5\ \mu s$, PRR = 1mHz, $Z_{out} \approx 50\Omega$
2. All diodes are 1N3064
3. $t_{pd} = \frac{t_{PHL} + t_{PHL}}{2}$
4. C_L includes probe and jig
5. When testing the SN5400/SN7400, connect all unused inputs to 2.4 V
6. For N = 10, $C_L = 15$ pF and $R_L = 400\ \Omega$
7. $V_{CC} = 5$ V

Figure 3.19

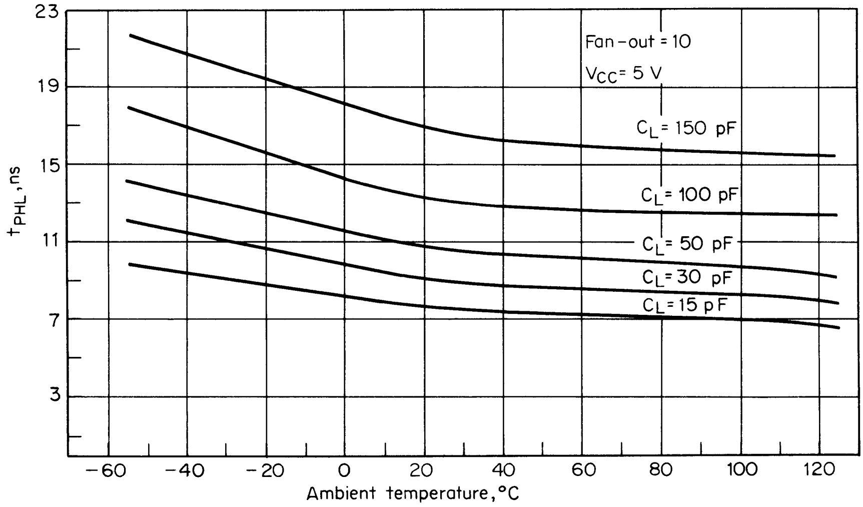


Fig. 3.20. Typical propagation delay time to logical 0, for SN5400.

are specified using the test circuit of Fig. 3.18, and typical values are shown in Figs. 3.15 and 3.16. All inputs are tied to ground, and maximum V_{CC} is used. Test limits are $|-20|$ mA $\leq I_{OS} \leq |-55|$ mA for Series 54 and $|-18|$ mA $\leq I_{OS} \leq |-55|$ mA for Series 74. Too large a value of I_{OS} could damage the device, and too small a value would degrade output logical 1 switching times, since load capacitance could not be charged rapidly.

Switching Speed. Two switching-time parameters are tested on TI 54/74 TTL gates: propagation delay time t_{PHL} (previously t_{pd0}) from a logical 1 to a logical 0 level at the output, and propagation delay time t_{PLH} (previously t_{pd1}) from a logical 0 to a logical 1 level at the output. Both parameters are measured with respect to the input pulse, using the test conditions of Fig. 3.19. The diodes, resistor R_L, and capacitor C_L are used to approximate the loading of 10 TTL gate inputs. These switching tests are performed at nominal conditions only: $V_{CC} = 5$ V, $T_A = 25°$C, and $N = 10$. Acceptable devices have $t_{PHL} \leq 15$ ns and $t_{PLH} \leq 22$ ns (SN5400/SN7400). Under these conditions a typical device will have $t_{PHL} \approx 7$ ns and $t_{PLH} \approx 11$ ns. Typical switching times as a function of ambient temperature and capacitive loading on the gate output are shown in Figs. 3.20, 3.21, and 3.22. Note that t_{PHL} decreases with increasing temperature, and t_{PLH} is essentially independent of temperature.

Typical propagation times as a function of supply voltage and capacitive loading on the output are displayed in Figs. 3.23, 3.24, and 3.25. Average propagation delay time t_{pd} is the numerical average of t_{PHL} and t_{PLH} under any given set of conditions.

An actual waveform of a 54/74 gate switching between the two logic levels is portrayed in Fig. 3.26. These pictures were taken with actual 54/74 gates tied to the output under test. Increased rise and fall times for the larger fan-outs are due primarily to the increased capacitance present.

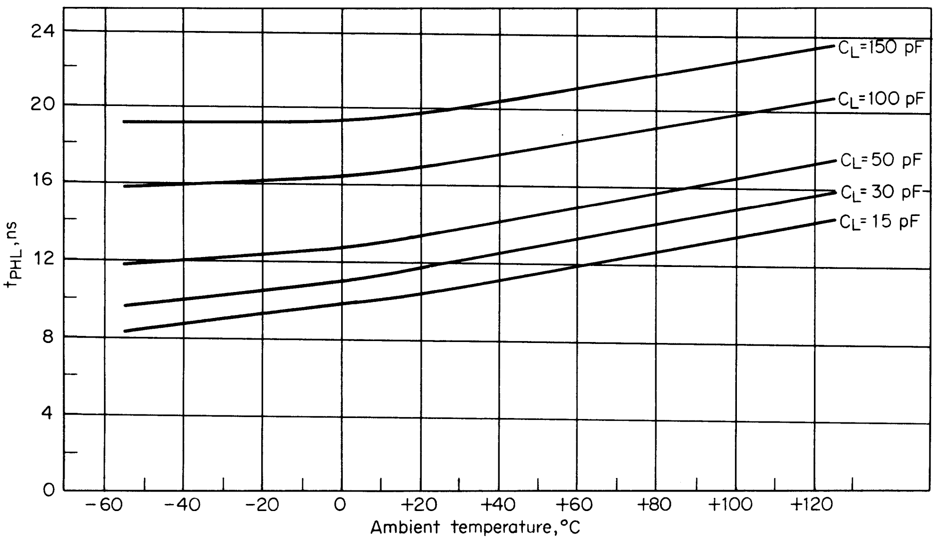


Fig. 3.21. Typical propagation delay time to logical 1, for SN5400.

Supply Current Characteristics. Power-supply current requirements for all Series 54/74 circuits are specified as maximum current drains with maximum permissible power-supply voltage V_{CC}. Test conditions are illustrated in Fig. 3.27. These are worst-case static d-c tests and are performed separately for the output at a logical 0, I_{CCL} (previously $I_{CC(0)}$) and for the output at a logical 1, I_{CCH} (previously $I_{CC(1)}$).

Since 54/74 gate packages contain different numbers of gates (one, two, three, four, or six), the supply current required by different gate packages will not be

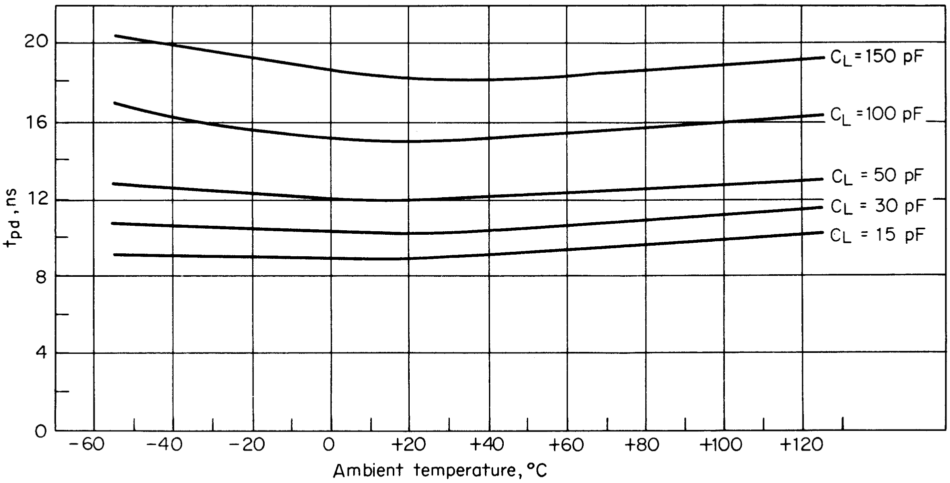


Fig. 3.22. Typical propagation delay, average of t_{PLH} and t_{PHL}, for SN5400.

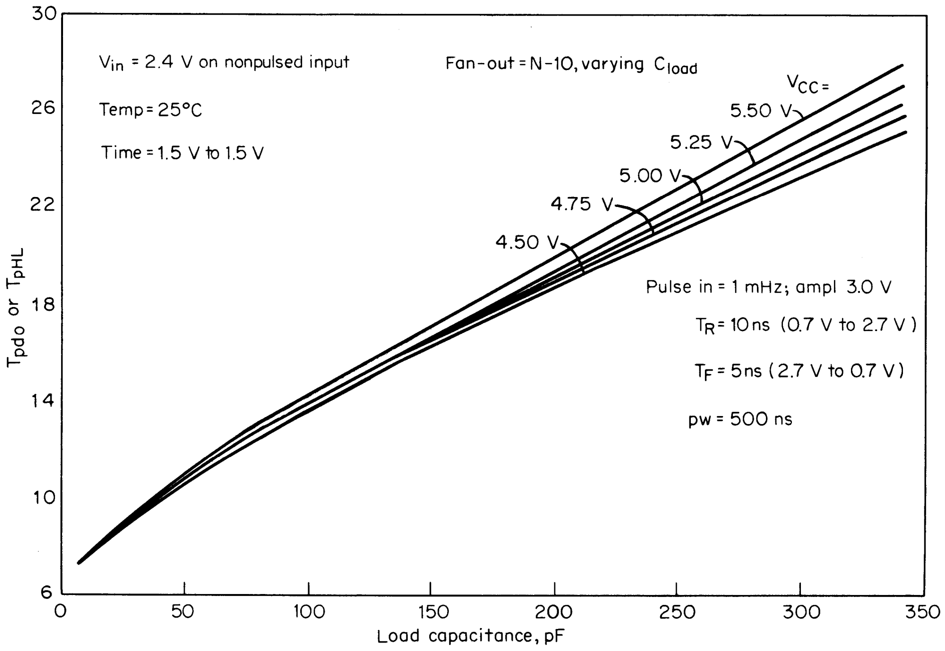


Figure 3.23

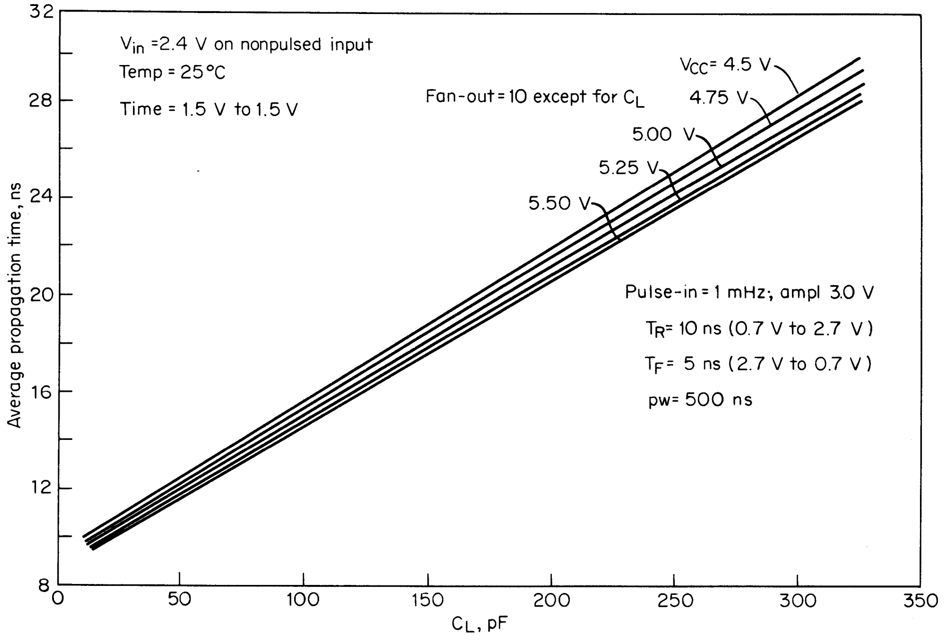


Figure 3.24

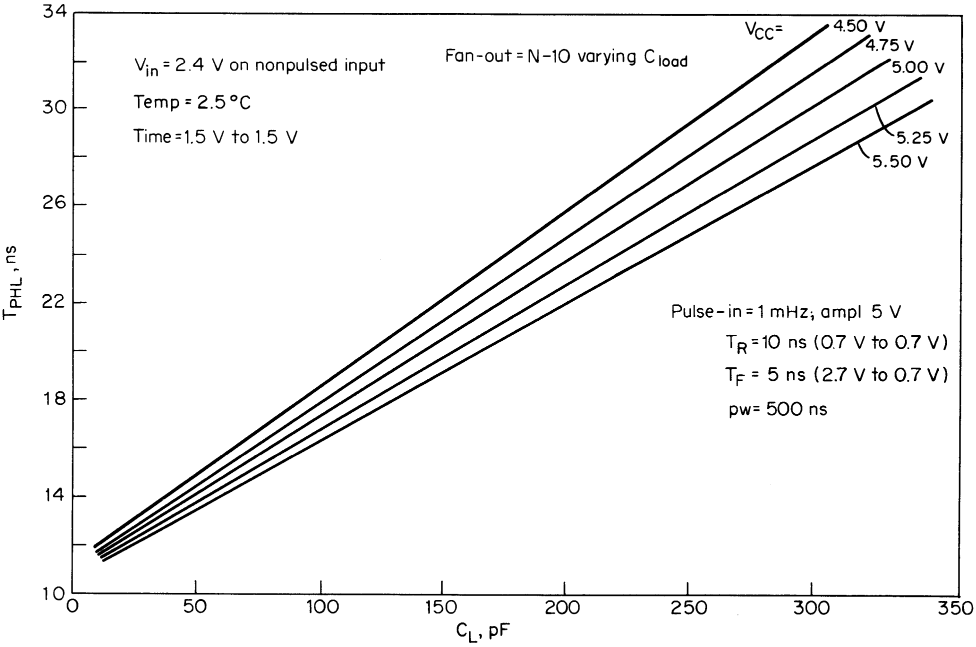


Figure 3.25

constant. However, the average *per-gate* supply current will be essentially the same in different device types. A normalized average per-gate value of supply current can be obtained for any gate package by dividing the total supply current (I_{CC}) by the number of gates the circuit contains. Of course, all gates within the package must be in the same logic state to make this a valid test.

Maximum I_{CCL} per gate is specified at 5.5 mA. Maximum I_{CCH} per gate is specified

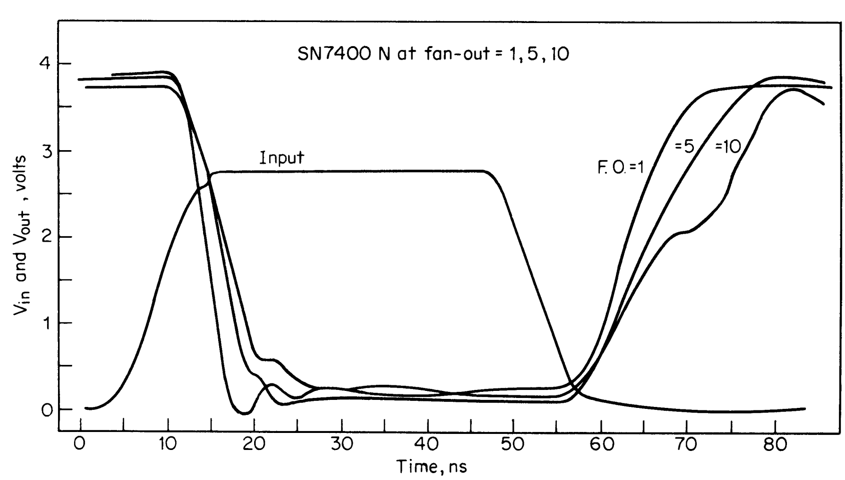
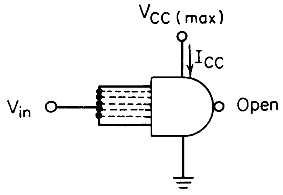


Fig. 3.26. Typical SN54/74 switching times at 25°C.



1. Logical 0 and logical conditions are tested
2. $V_{in} = 5$ V for logical 0 test
3. $V_{in} = 0$ for logical 1 test
4. $V_{CC} = 5.5$ V for series 54
5. $V_{CC} = 5.25$ V for series 74
6. All gates are tested simultaneously

$$\text{Average per-gate value} = \frac{I_{CC}\text{total}}{\text{no. of gates in package}}$$

Fig. 3.27. Worst-case testing for logical 0 and logical 1 supply current.

as 2.0 mA. Worst-case supply current tests are performed at maximum supply voltage. At the nominal supply voltage of 5 V, typical I_{CCL} per gate is 3 mA, and typical I_{CCH} is 1 mA. Thus I_{CCL} is about three times as large as I_{CCH}. Figure 3.28 illustrates the variation of supply current with supply voltage. The complex variations of supply current with ambient temperature shown in Fig. 3.29 illustrate the normal opposing temperature coefficients present in an integrated circuit.

When power-supply currents are specified by a manufacturer at the nominal V_{CC} (5 V), the true worst-case current drain is not obtained. I_{CC} at nominal V_{CC} is, in fact, approximately 91 percent of the worst-case value, and the resulting power dissipation is approximately 82 percent of the worst-case value.

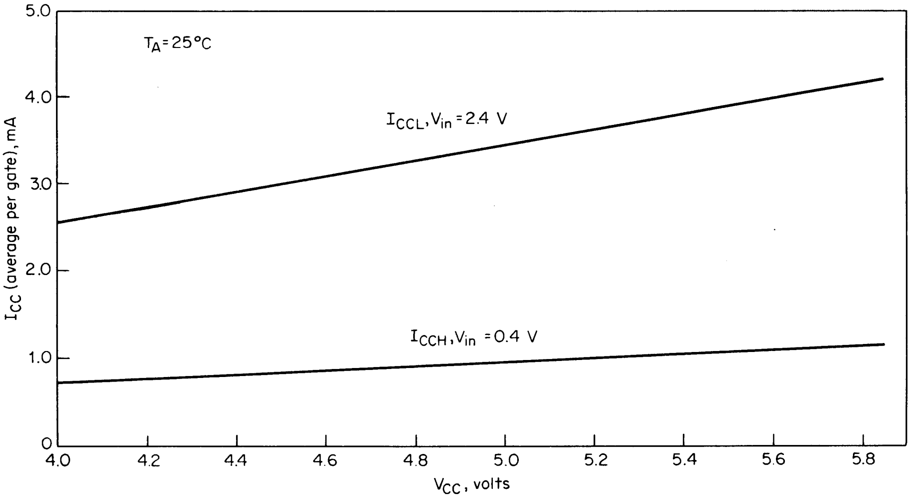


Figure 3.28

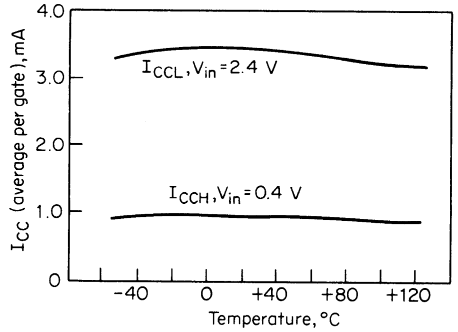


Figure 3.29

The currents mentioned above are for static d-c conditions only. When a 54/74 TTL gate switches from one logical state to the other, a momentary increase in supply current drawn by the gate occurs. With no load on the output, this increased supply current is used primarily to charge internal capacitance and lasts only a few nanoseconds. Any capacitance connected to the output must be charged and discharged as the gate output switches. When an output switches from logical 0 to logical 1, any capacitance connected to the output must be charged by increased I_{CCH} current. This charging current may be as much as the I_{OS} of the gate. Figure 3.30 illustrates the effect of gate switching frequency on average supply current (and thus power dissipation of the gate). Note the pronounced effect of load capacitance on average supply current.

D-C Noise Margin. D-c noise margin is defined as the difference between the guaranteed logic state voltage limits of a driving gate and the voltage requirements of a driven device. The 400-mV area of safe operation commonly called *guaranteed d-c noise margin* is illustrated in Fig. 3.31. The principal motivating factor in setting up the test conditions is to guarantee the user an absolutely safe operating margin, as pointed out in the earlier discussion on d-c testing of TTL gates.

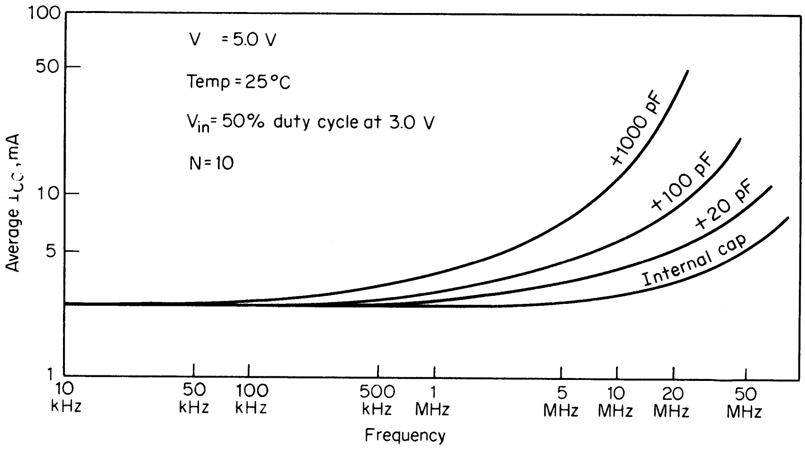


Figure 3.30

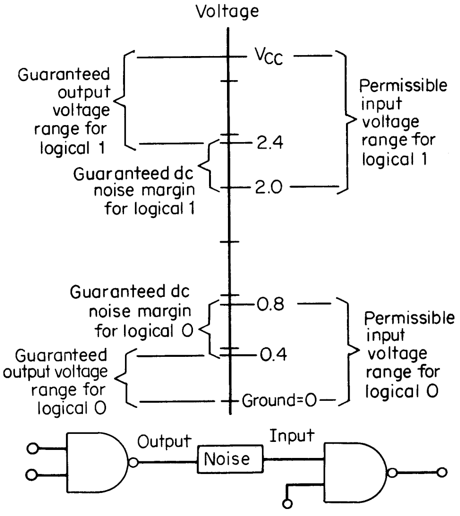

Voltage
Guaranteed output voltage range for logical 1
Vcc
2.4
2.0
Permissible input voltage range for logical 1
Guaranteed dc noise margin for logical 1
Guaranteed dc noise margin for logical 0
0.8
0.4
Ground=0
Guaranteed output voltage range for logical 0
Permissible input voltage range for logical 0
Output
Noise
Input

Fig. 3.31. Definition of noise margin for SN54/74 gates.

TTL gate outputs are tested under worst-case temperature and V_{CC}, and they are guaranteed not to exceed 0.4 V for a logical 0 or fall below 2.4 V for a logical 1. However, when voltages are applied to the input to obtain these output conditions, 2.0 V is used as a logical 1 input and 0.8 V is used as a logical 0. A guaranteed worst-case margin of 400 mV for both logic states results (see Fig. 3.31). In the logical 0 state, the output cannot go above 0.4 V, but the driven gate requires an input voltage higher than 0.8 V before significant output voltage change occurs. A 400-mV noise excursion is then possible without propagating the noise through subsequent stages.

Although only a 400-mV noise margin is guaranteed, a TTL gate typically exhibits a noise margin in excess of 1.0 V. Series 54/74 gates change state as the input voltage passes through a threshold voltage V_T of about 1.35 V ($T_A = 25°\text{C}$). The output is typically 3.3 V in the logical 1 state (see Fig. 3.4) and 0.2 V in the logical 0 state. Therefore, in the logical 1 state the output can typically tolerate 1.5 V of negative-going noise on the line connecting the two gates before causing the driven gate to change state falsely. Similarly, 1.15 V of positive-going noise can typically be tolerated on a line in the logical 0 state. As shown in Fig. 3.32, TTL gates have more than 1-V typical noise margin with V_{CC} of 5.0 V in both logic states. Note that the margin varies with a change in ambient temperature but is never less than 400 mV.

A-C Noise Margin. The term *d-c margin* seems a little inappropriate when applied to noise, which by general conception is an a-c factor. While “d-c noise” is not impossible, the user is much more concerned with a transient type of noise. In circuits with the speed of integrated-circuit logic, microsecond-wide pulses are extremely long and may be treated as d-c so far as operating margins are concerned. As the pulse widths shorten into the low-nanosecond region, a limit is reached where an input pulse can be shorter than the time required for a signal to propagate through the device. As this point is approached, higher-amplitude pulses are required to

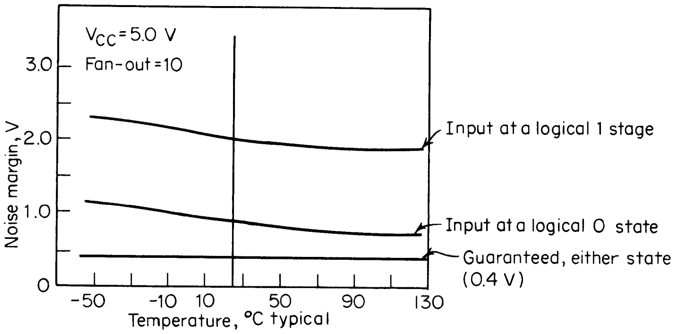


Fig. 3.32. SN54/74 typical d-c noise margins.

effect a change at the output, and eventually any reasonable-amplitude pulse will not be propagated through. There is very little of the delay-line effect involved; where once a pulse is introduced, it will propagate through the line, even though the line delay is many times longer than the pulse. Rather, it is a matter of changing the input to a different state long enough for the device to respond. This effect accounts for the sharp knee in the a-c noise-immunity curve of Fig. 3.33.

There is, however, some effect traceable to the pulse amplitude. This may be broken into two separate reactions:

1. Effect of the input being driven into the opposite state marginally or completely. This affects the propagation delay of the device.
2. Capacitive storage and coupling effects which cause high-amplitude pulses to enter and be partially stored. This behavior allows the gate to continue to respond to the pulse even after it has disappeared at the input. There can also be a progressive reaction to closely repeated high-amplitude pulses. Although the first pulses of a sequence may not trigger gate response, subsequent ones may. If the time between pulses is greater than the pulse

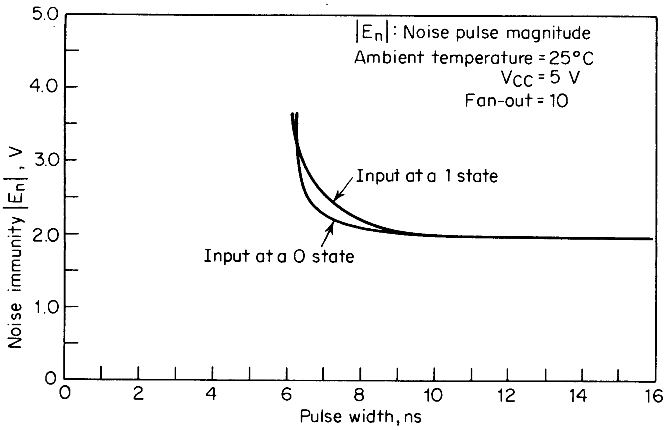


Fig. 3.33. SN54/74 typical a-c noise immunity.

width, there should be no buildup. Again, this applies only to very fast, very short high-amplitude pulses.

In the pulse width/amplitude noise-immunity curves of Fig. 3.33, only the worst case is considered, which is a 0-to-1 noise transition. Immunity to 1-to-0 transitions of the NAND gate is generally better because of the longer propagation time of the device output to the 1 state.

Noise Rejection. The ability of a logic element to operate in a noise environment involves more than the d-c or a-c noise margins. To be a problem, an externally generated noise pulse must be received into the system and cause a malfunction. Stable logic systems with no storage elements are practically impervious to a-c noise, although large d-c voltages could cause trouble. Systems with triggerable storage elements, or those operating fast enough for the noise to appear as a signal, are much more susceptible.

The noise voltage must be introduced into the circuit by radiated or coupled means. The amount of noise power required to develop a given voltage is strictly a function of the circuit impedance. It is in this way that the low output impedances of TTL improve noise immunity. Noise power must be transferred from the noise source, with some arbitrary impedance, through a coupling impedance to the impedance of the circuit under consideration.

A typical circuit where the coupling impedance is stray capacitance, and the load impedance is provided by gates, is shown in Fig. 3.34. In relatively tight coupling such as this, the loading effect on the driving source is significant enough to be considered. However, since the source effect is difficult to assess and is in a direction to improve rather than degrade the noise rejection, its effects will be ignored, resulting in a worst-case type of response indication. Also, in the case of radiated noise, the source resistance is definitely a factor in noise coupling, essentially replacing the reactive coupling impedance.

Thus ignoring the driving source impedance to make the conditions more nearly standard, it is possible to determine a set of curves relating the developed noise pulse to the noise source amplitude, the noise rise (or fall) time, the coupling impedance, and the load impedance. Such curves have been developed by Elmore and Sands† for several different input waveforms. The type that is most applicable

---

†W. C. Elmore, and M. Sands, "Electronics Experimental Techniques," pp. 30 ff., McGraw-Hill Book Company, New York, 1949.

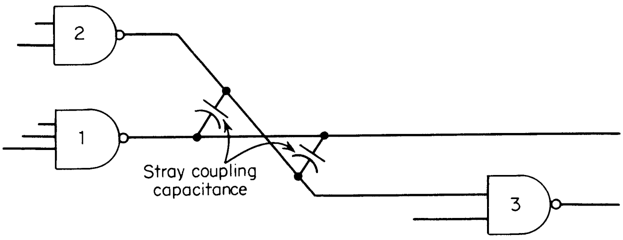


Figure 3.34

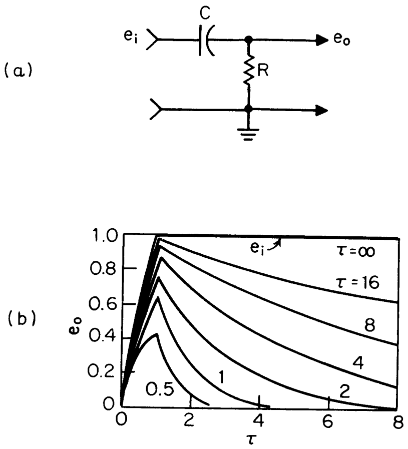


Figure 3.35

here is the ramp input, since the TTL wavefront is essentially a ramp with a *dv/dt* of 0.4 V per ns to 0.8 V per ns, considering the extremes between SN54H/74H and SN54L/74L.

Figure 3.35*a* shows the equivalent circuit from which the plot of Fig. 3.35*b* was developed. The input pulse (shown as a heavy line) is a step signal with a linear rise, requiring unit time (normalized), and the output pulse is represented analytically by

$$e_0 = \tau(1 - e^{-t/\tau})$$

with

$$\tau = RC$$

holding for unit time. This is followed by an exponentially decaying voltage having a time constant τ. Values of τ and t on the figure are normalized by the value of the total rise time of the simulated noise pulse e_i. Using Fig. 3.35*b*, it is possible to determine the pulse width and amplitude of the coupled noise pulse.

As an example, allow a pulse of 4-V amplitude rising at 0.4 V per ns with gate 2 output (Fig. 3.34) in the logical 1 state. Assume the nominal 70-Ω output impedance and 70 pF of coupling capacitance.

$$\tau = RC = (70 \times 10^{-12})(70) = 4.9 \times 10^{-9} = 4.9 \text{ ns}$$

$$\text{Total rise time} = \frac{4 \text{ V}}{0.4 \text{ V/ns}} = 10 \text{ ns}$$

Now to convert the normalized values of τ and t of Fig. 3.35*b* to actual values, multiply by 10 ns. The output voltage scale will be multiplied by 4 V. Using the $\tau = 0.5$ (actual value $\tau = 5$) curve gives a peak e_0 of 1.64 V (0.41 × 4) and a pulse width of 12 ns (1.2 × 10) at the 50 percent points. To determine whether or not this pulse will cause interference, these values (1.6 V and 12 ns) are entered on the graph of Fig. 3.33. Since the gate has 2.0 V of noise immunity at this point, the gate will not be affected.

If an open-collector gate such as the SN5403/SN7403 were to be used with a 1-kΩ pull-up resistor (or other resistive pull-up device, such as an RTL or DTL), the picture would change:

$$\tau = (70 \times 10^{-12})(1 \times 10^{3}) = 70 \times 10^{-9} = 70 \text{ ns}$$

Now the amplitude (from the curves) is about 3.6 V (0.9 × 4), and the pulse width at the 50 percent points is approximately 60 ns (6 × 10). This will certainly cause interference to gate 3 (Fig. 3.34).

This is obviously an oversimplification. The coupling impedances are complex (but resolvable into RLC series coupling elements), and the gate output impedance changes with load. But our purposes here are to show why and how the low impedance of TTL rejects noise and to facilitate a comparison with other logic types.

The ability to operate in a noisy environment is, then, an interaction of the built-in operating margins, the time required for the device to react, and the ease with which a noise voltage is developed. In all except the ability to react to short noise pulses, if developed, TTL design has emphasized noise rejection.

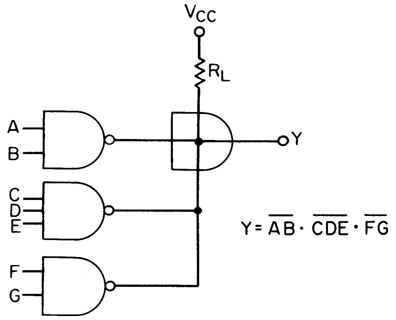
So far nothing has been said about noise in devices other than gate circuits. Many MSI devices are complex gate arrays, and because of their small size, they are superior in noisy-environment operation to their discrete gate equivalents. Noise tolerance of latching devices is implied in the setup times, hold times, clock pulse width, data pulse widths, and similar parameters. Output impedances and input noise margins are quite similar to those of the gates and may be treated similarly. A point to be remembered is that if a latching device does become noise-triggered, the effective error is stored and does not disappear with the noise.

3.4 CIRCUIT CHARACTERISTICS OF SPECIALIZED GATES

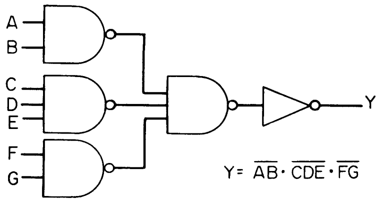
In the preceding discussion, the NAND gate was used as the basic model for characterizing 54/74 TTL. The basic electrical parameters developed for the NAND gate can be applied to all SSI and MSI functions with few exceptions. The gates discussed in this section are variations on the basic TTL structure, and the particular differences in circuit structure and parameters are pointed out.

Open-collector Gates, Inverters, and Drivers. As previously explained, the totem-pole output configuration has the advantage of low output impedance in both logical states and is largely responsible for the capability to drive highly capacitive loads, the low-noise susceptibility and the high-speed performance characteristics of Series 54/74. This output circuit configuration, however, is not well suited to the wire-AND logic connection.

Wire-AND-ing means simply tying gate outputs together (Fig. 3.36*a*) to obtain the AND function. In comparison with the method of using three NAND gate levels (Fig. 3.36*b*) to obtain the NAND-AND function, this saves two levels of logic and their associated delays. In the wire-AND configuration, there is a very good possibility that a single gate will have a logical 0 output, and all others will be at a logical 1. If the wire-AND connection were implemented using gates with a totem-pole output stage, this single output stage in the 0 state would be required to sink current


Vcc
RL
A
B
C
D
E
F
G
Y
Y= $\overline{AB}\cdot\overline{CDE}\cdot\overline{FG}$

(a) Wire-and connection


A
B
C
D
E
F
G
Y
Y= $\overline{AB}\cdot\overline{CDE}\cdot\overline{FG}$

(b) Connection using gates with totem-pole outputs

Fig. 3.36. Comparison of open-collector gates in wire-AND configuration with totem-pole output gates.

from the active pull-up transistors of all the other gates. Apart from degradation of the logical 0 voltage level, there exists the possibility of high-current damage to the output stage transistors. A totem-pole output can source as much as 55 mA (I_{OS}—only one gate output in a package should be short-circuited), and a gate output is guaranteed to sink only 16 mA (I_{OL}). It is for these reasons that the open-collector versions exist and that gates with totem-pole outputs are not generally recommended for use in the wire-AND configuration.

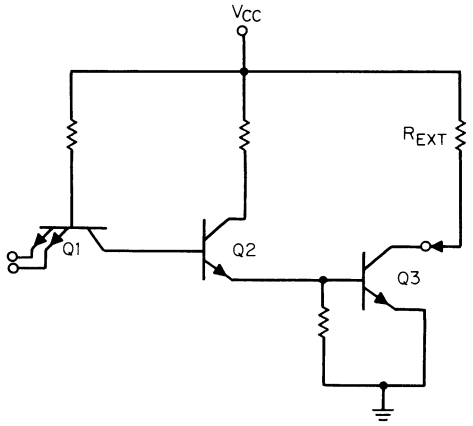

Vcc
REXT
Q1
Q2
Q3

Fig. 3.37. Open-collector NAND gate showing external resistor connection.

Table 3.3 is a summary of SSI and MSI logic functions with open-collector output stages that might possibly be used in the wire-AND configuration. The parameters most used in the design of a wire-AND circuit are shown. Note that the SN54/7426 presently has the lowest guaranteed I_{OH}. The other tables and calculations use the most common specifications, those of the SN54/7401; equations and examples are given so that calculations can be made for any of the devices.

Figure 3.37 is a typical output stage circuit diagram for these devices. Some thought must be given to the value of the pull-up resistor R_L. A large value for R_L will reduce power consumption, but in the logical 1 state it also determines output impedance. Since high output impedance tends to increase propagation delay and increase noise susceptibility, a compromise is necessary between switching time and power consumption—and fan-out also must be considered. A maximum value for R_L must be found which will ensure that the output will be equal to or greater than 2.4 V in the logical 1 state. For the logical 0 state a minimum value for R_L must

Table 3.3. Guaranteed Ratings for Output Stage of Series 54/74 Open-collector Logic Functions

Type	Logic function	V_{OH} max, V	I_{OH} at V_{OH} max, μ A	V_{OL} max at I_{OL}, V	I_{OL}, mA
Standard SSI					
SN54/7401	Quad 2-input NAND	5.5	250	0.4	16
SN54/7403	Quad 2-input NAND*	5.5	250	0.4	16
SN54/7405	Hex inverter	5.5	250	0.4	16
SN5406	Hex inverter buffer/driver	30	250	0.7/0.4	30/16
SN7406	Hex inverter buffer/driver	30	250	0.7/0.4	40/16
SN5407	Hex buffer/driver	30	250	0.7/0.4	30/16
SN7407	Hex buffer/driver	30	250	0.7/0.4	40/16
SN54/7409	Quad 2-input AND	5.5	250	0.4	16
SN5416	Hex inverter buffer/driver	15	250	0.7/0.4	30/16
SN7416	Hex inverter buffer/driver	15	250	0.7/0.4	40/16
SN5417	Hex buffer/driver	15	250	0.7/0.4	30/16
SN7417	Hex buffer/driver	15	250	0.7/0.4	40/16
SN54/7426	Quad 2-input NAND	15/12	1,000/50	0.4	16
High-speed					
SN54/74H01	Quad 2-input NAND	5.5	250	0.4	20
SN54/74H05	Hex inverter	5.5	250	0.4	20
SN54/74H22	Dual 4-input NAND	5.5	250	0.4	20
MSI					
SN54/7445	BCD-to-decimal decoder/driver	30	250	0.9/0.4	80/20
SN54/74145	BCD-to-decimal decoder/driver	30	250	0.9/0.4	80/20
SN54/74156	Dual 2-line-to-4-line decoder	5.5	250	0.4	16
SN7488	256-bit read-only memory	5.5	100	0.4	12

*Same pin configuration as SN54/7400.

be found which will ensure that V_{OL} will not exceed 0.4 V with the combined load of the resistor and the fan-out to other gate inputs.

Note that the values chosen for V_{OH} ($\geq$2.4 V) and V_{OL} ($\leq$0.4 V) are the guaranteed output logic levels for a standard totem-pole output stage. Using these values for the wire-AND circuit design gives a guaranteed d-c noise margin of 400 mV, as previously discussed. The design could also be based on values nearer to the gate input parameters V_{IL} ($\geq$0.8 V) and V_{IH} ($\leq$2 V), with the understanding that the d-c noise margin will be reduced accordingly. R_L, for either logic state, may be calculated from

$$R_L = \frac{V_{RL}}{I_{RL}}$$

where V_{RL} equals the voltage across R_L, and I_{RL} equals the current through R_L.

Figure 3.38 shows the case for a logical 1 output at the wire-AND node and the currents that must be considered. Since at least 2.4 V should be present, no more than 2.6 V (see Fig. 3.38) can be dropped across R_L. The current through R_L is a composite of current into the loads ϵI_{IH} and leakage current into output transistors which are biased off ϵI_{OH}. Both I_{OH} and I_{IH} are data sheet specifications; they are 250 μA maximum and 40 μA maximum, respectively. The maximum value of R_L is found in this case; a greater value would result in a lower logical 1 output voltage level.

A minimum value for R_L is found by considering the logical 0 conditions (see Fig. 3.39). Again R_L is permitted to drop a maximum voltage as dictated by noise margin, i.e., supply voltage minus the maximum allowable logical 0 level (refer to Fig. 3.39). Current through R_L and the gate loads must now be limited to the

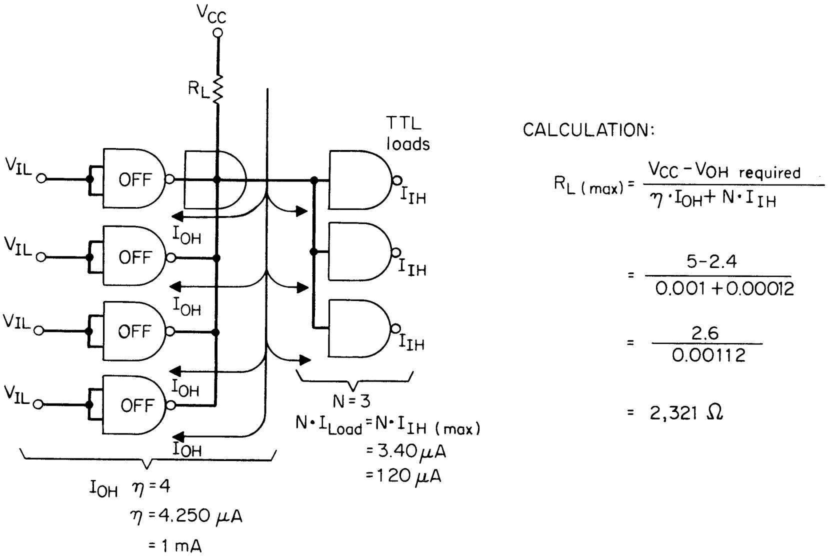
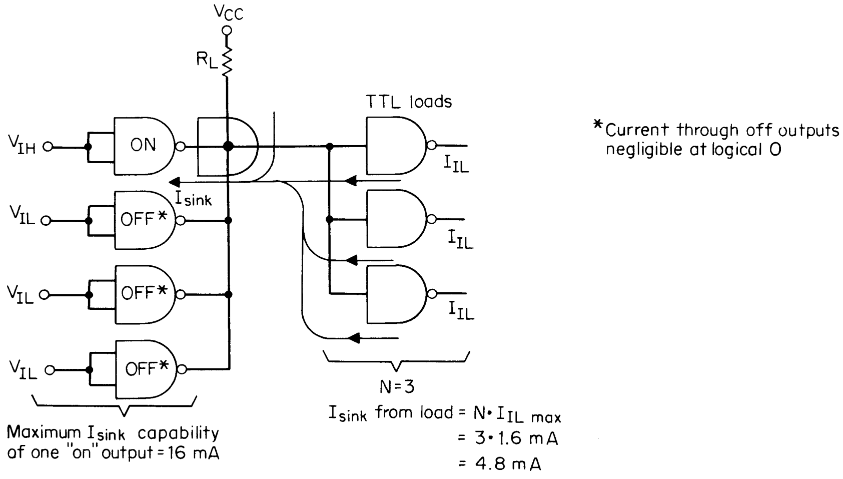


Fig. 3.38. Logical 1 circuit conditions, calculation of $R_{L(\max)}$.



V_{CC}
R_L
TTL loads
V_{IH}
ON
V_{IL}
OFF*
I_{sink}
V_{IL}
OFF*
V_{IL}
OFF*
I_{IL}
I_{IL}
I_{IL}
*Current through off outputs negligible at logical 0
Maximum I_{sink} capability of one "on" output = 16 mA
N=3
I_{sink} from load = $N \cdot I_{IL\ max}$
= $3 \cdot 1.6$ mA
= 4.8 mA

Calculations:

$$R_{L(min)} = \frac{V_{CC} - V_{OL\ \text{required}}}{I_{sink}\ \text{capability} - I_{sink}\ \text{from TTL loads}}$$

$$= \frac{5.0 - 0.4}{0.016 - 0.0048}$$

$$= \frac{4.6}{0.0112}$$

$$= 410\ \Omega$$

Fig. 3.39. Logical 0 circuit conditions, calculation of $R_{L(\min)}$.

maximum sink-current capability of one output transistor. If several gates are wire-AND connected, current through R_L may be shared by parallel output transistors. However, unless it can be guaranteed that more than one transistor will be on during logical 0 periods, the current must be limited to one I_{OL}. It is essential to note, however, that logical 1 output impedance due to the pull-up resistor is significantly greater than that of the active pull-up totem-pole output. This characteristic of an open-collector type of device restricts its use in applications that are critical with respect to switching speed, a-c noise immunity, and high-capacitive drive. Table 3.4 provides minimum and maximum resistor values for several combinations of TTL loads and wire-AND connected output stages.

With a fan-out of 10 and no wire-AND connections, the calculation for R_L indicates an infinite resistance value ($V_{RL} \div 0 = \infty$). However, a 4-kΩ resistor satisfies the logical 1 condition, and it limits the logical 0 output voltage to less than 0.43 V.

Since $R_{L(\max)}$ decreases with increasing η, the maximum allowable η is reached when R_L decreases to the value of $R_{L(\min)}$. Table 3.4*b* shows the maximum value of η for the value of $R_{L(\min)}$ associated with fan-outs of 1 through 10.

Compare the SN54/7426, which is similar to the SN54/7401 with the exception that it has a guaranteed I_{OH} of 50 μA. To see the improvement in η that could

Table 3.4

(*a*)

Fan-out to TTL loads, N	Maximum η*						
	1	2	3	4	5	6	7
1	8,965	4,814	3,291	2,500	2,015	1,688	1,452
2	7,878	4,482	3,132	2,407	1,954	1,645	1,420
3	7,027	4,193	2,988	2,321	1,897	1,604	1,390
4	6,341	3,939	2,857	2,241	1,843	1,566	1,361
5	5,777	3,714	2,736	2,166	1,793	1,529	1,333
6	5,306	3,513	2,626	2,096	1,744	1,494	1,306
7	4,905	3,333	2,524	2,031	1,699	1,460	1,280
8	4,561	3,170	2,429	1,969	1,656	X	X
9	4,262	3,023	X	X	X	X	X
10	4,000†	X	X	X	X	X	X

(*b*)

Fan-out to TTL loads, N	R_L, minimum ohms	η,* maximum
1	319	32
2	359	28
3	410	24
4	479	21
5	575	17
6	718	13
7	958	9
8	1,437	5
9	2,875	2
10	4,000†	1

*η is the number of open-collector outputs tied to the wire-AND node.
X Not recommended or not possible.
†The theoretical value is ∞. See explanation in text.
All values shown in the table are based on $V_{CC} = 5$ V.

Logical 1 conditions: V_{OH} required $= 2.4$ V
Logical 0 conditions: I_{OH} maximum $= 250\ \mu$A
V_{OL} required $= 0.4$ V
I_{OL} maximum $= 16$ mA

Equation for calculation of $\eta_{\max}$:

$$\eta_{\max} = \frac{2.6}{R_{L(\min)} \cdot I_{OH}} - \frac{N \cdot I_{IH}}{I_{OH}}$$

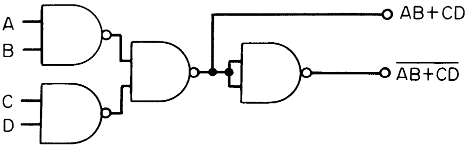


Figure 3.40

be made, consider the specifications of the SN54/7426, which include an I_{OH} of 50 μA rather than 250 μA previously used. For a fan-out of one gate load using the SN54/7426, η can be a maximum of 162.

AND Gates. As a further aid in system cost minimization and control of critical propagation delays, some available devices include an additional inverting stage. The delay introduced by this technique is less; it is more predictable than cascading two gates to obtain a noninverting output, and the power drain is less. This is attained by performing the inversion at low power levels and eliminating stray capacitance and inductance effects from outside. Less than 5 mW per gate increase in power is required, with about 4 ns propagation delay increase over the NAND gates. Otherwise, input and output parameters are identical.

This function is available in the triple 3-input and dual 4-input high-speed gates, SN54H11/74H11 and SN54H21/74H21, respectively, and the SN5408/7408 quadruple 2-input. An open-collector version of the SN5408/7408 is available as the SN5409/7409.

NOR Gates. AND logic is easily obtained from the basic NAND gate simply by inverting the output. NOR and OR functions, however, require significant additional logic levels if implemented strictly from NAND gates (see Fig. 3.40). Propagation delay, cost, and size may be seriously affected if this scheme is used repeatedly. For this reason a quadruple 2-input NOR gate, the SN5402/7402, is included in Series 54/74. The schematic diagram is shown in Fig. 3.41. In this circuit if either transistor *Q2A* or *Q2B* conducts, *Q3* will be on and *Q4* off, i.e., an output logical 0. *Q2A* and *Q2B* conduct when the associated input is at a high level; hence the resultant NOR function $Y = \overline{A + B}$.

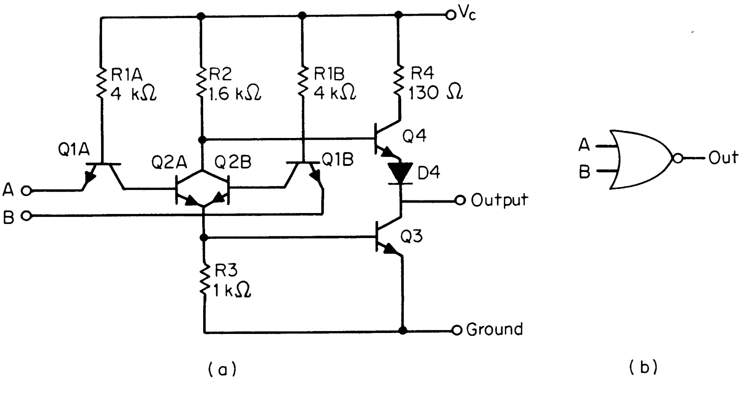


Fig. 3.41. Circuit and logic schematic of 2-input NOR gate SN5402/7402.

Use of the totem-pole output, equivalent transistor geometries, and equivalent resistor values render the performance characteristics of this device equal in all respects to that of the standard SN5400/7400 NAND gate.

AND-OR-INVERT and Expander Gates. Although wire-AND performing the function

$$Y = \overline{AB} \cdot \overline{CD} = \overline{AB + CD}$$

is possible with the TTL open-collector NAND gates, a penalty is always paid in loss of fan-out and, if power is minimized, in loss of speed. For these reasons, the AOI (AND-OR-INVERT) TTL gates were introduced to perform the same function in a single stage, thus maintaining the 15-ns maximum delay and full fan-out of 10. These are available in the standard Series 54/74 and in the low-power and high-speed versions. The following logic functions are available:

$$Y = \overline{AB + CD + X}$$

$$Y = \overline{AB + CD + EF + GH + X}$$

$$Y = \overline{AB + CD + EF + GHI + X}$$

$$Y = \overline{ABCD + EFGH + X}$$

$$Y = \overline{ABC + DEF}$$

$$Y = \overline{ABC + DE + FG + HIJ}$$

$$Y = AB + CD + EF + GHI + X \text{ (noninverting)}$$

The term X indicates the expandable input which is available on every function except those of the low-power line, although nonexpandable versions are available in each function. Expander gates are available with various input arrangements; it is important to note, however, that not all expanders can be used with all expandable gates. The acceptable combinations are stated in the data sheets. The number of expanders that can be attached to any gate is limited; this is also specified in the data sheets.

Propagation delays are stated for the AOI gates without expanders attached. When expanders are attached, propagation delay increases, owing to capacitive loading of the AND-ing collector resistor. Propagation delays are stated in the expanded conditions for the AOI gates only under the expander parameters, since propagation delay of the expander is measured in conjunction with the AOI gate. Since there is an increase of 5 ns in t_{PHL} and an increase of 8 ns in t_{PLH} caused by the addition of one expander to the SN5450/SN7450 or SN5453/7453, similar increases should be allowed for each expander connected. Calculation using the 1.6-kΩ collector resistor in these devices indicates about 1 ns per pF; this indicates a typical capacitance addition of about 8 pF per expander connected.

The high-speed expanders, for use with SN54H/74H expandable gates, do not specify a propagation delay, but they do specify a capacitance of 1.3 pF at the output. Since this capacitance is quite small in relation to the stray capacitance in normal interconnections, it is of little consequence. A more useful concept is that of the expander node resistance and stray capacitance. Since the resistor value is half that of the SN5453/7453, about half the increase in delay is to be expected, i.e., 2.5 ns t_{PHL} and 4 ns t_{PLH} per expander attached.

It is possible to mix the standard AOI gates and expanders with those of the high-speed line in certain restricted arrangements. These are indicated in Table 3.5.

It is practical to use the expansion point of the AOI gates to introduce delay into the circuit by adding shunt capacitance to ground. Between 0.5 and 1 ns per pF will be achieved. The delay is obtained by changing the rise and fall time of the output. Consequently, it may be necessary to reshape the waveform after long delays have been achieved.

Schmitt Input NAND Gates. Various problems are encountered when the inputs to logic devices change state slowly:

1. Rise-time-sensitive devices may not operate.
2. Instability may result if logic gates are biased in the active region too long.
3. Propagation delays become difficult to predict. Many logical decisions may be made while one data input is in transition. Timing becomes more difficult.

For these reasons, and since many data sources are inherently slow, a device to "square up" the waveforms is desirable. The SN54/7413 Schmitt trigger provides this shaping by introducing positive feedback into a circuit to obtain high gain and hysteresis. The SN54/7413 is a dual 4-input Schmitt trigger fully compatible with TTL or DTL, and it requires only the standard V_{CC} of +5 V to operate. Figure 3.42 shows the V_{in}/V_{out} characteristic of the SN54/7413, and the 800 mV of hysteresis present. Note that were the poor waveform to be fed directly into logic devices,

Table 3.5

AOI gate	Expander	Remarks
SN5450/7450 SN5453/7453	SN5460/7460 SN54H60/74H60 SN54H62/74H62	See data sheet $I_{IL} = 2$ mA $I_{IL} = 2$ mA; t_{PD} less than for 4 SN5460/7460 expanders
SN54H50/74H50 SN54H53/74H53 SN54H55/74H55	SN54H60/74H60 SN54H62/74H62 SN5460/7460	See data sheet See data sheet Drive capability adequate, but not guaranteed; slight speed loss compared with SN54H50-54H60; speed gain compared with SN5450-SN5460
SN54H52/74H52	SN54H61/74H61	See data sheet (Neither device should be combined otherwise than as shown here.)
SN54L51/74H51 SN54L54/74L54 SN54L55/74L55	None None None	

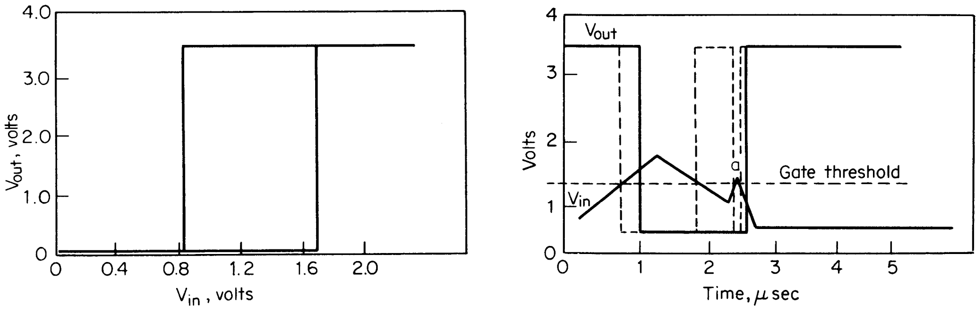


Fig. 3.42. Typical transfer characteristics.

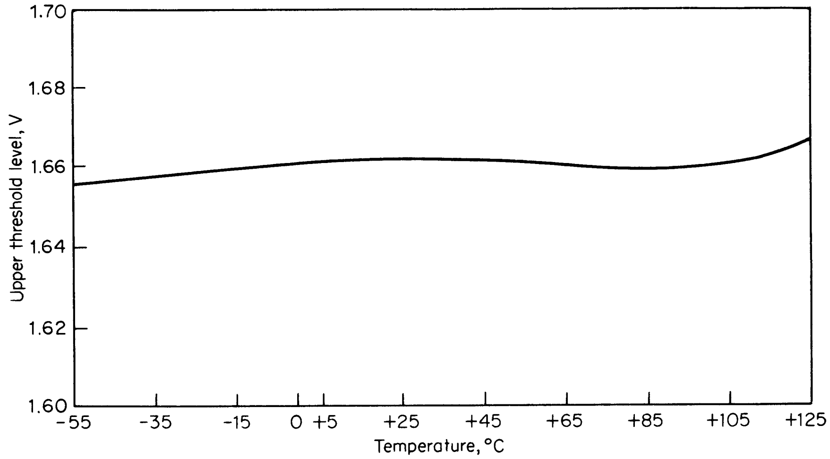


Fig. 3.43. Typical temperature variation of upper threshold voltage, SN5413.

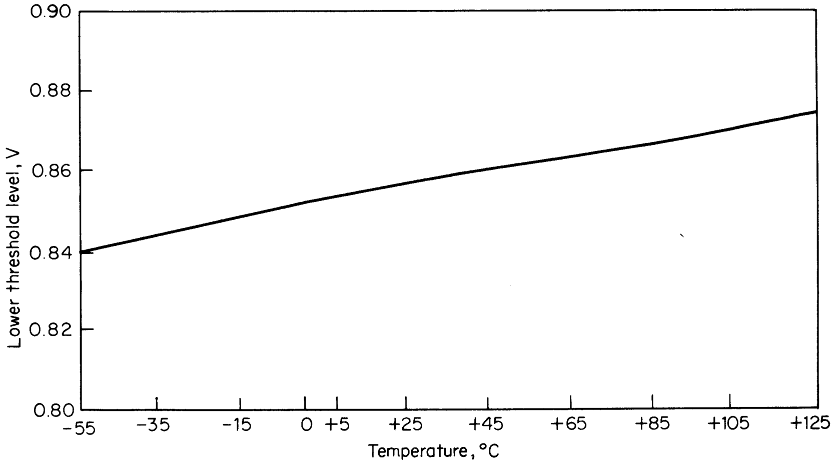


Fig. 3.44. Typical temperature dependence of lower threshold voltage, SN5413.

double triggering would occur at point *a* in the waveform. Dotted lines in Fig. 3.42 show a typical SN54/74 gate response to the input waveform.

Input currents are limited to -1.6 mA and 40 μA in the low and high states, respectively, to minimize loading on the signal source. However, these current requirements must be supplied by the signal source, and the source impedance is thus restricted. For example, if the source voltage can drive to no lower than -1 V, the 0-state impedance must be no greater than

$$Z_{SL} = \frac{V_{S(out)L} - V_{IL}}{I_{IL}}$$

$$= \frac{-1 - 0.8}{1.6} = \frac{1.8\text{ V}}{1.6\text{ mA}} = 1{,}125\ \Omega$$

Greater or smaller driving voltage would allow greater or smaller impedance in the source. A similar calculation can be made for the high state:

$$Z_{SH} = \frac{V_{S(out)H} - V_{IH}}{I_{IH}}$$

$$= \frac{V_{S(out)H} - 2.4}{40\text{ mA}}$$

The upper and lower trip points are stable over the full temperature range, showing a typical change of less than ±2.5 percent, which establishes a stable hysteresis characteristic. Curves showing the input parameter variations with temperature and V_{CC} are shown in Figs. 3.43, 3.44, 3.45, and 3.46.

Propagation delays are measured from the approximate input thresholds of 0.9 and 1.7 V to the output at 1.5 V. With standard TTL loads, propagation delays are $t_{PHL} = 30$ ns maximum and $t_{PLH} = 35$ ns maximum. Typical delays, including variation with V_{CC} and temperature, are shown in Figs. 3.47 and 3.48. With this

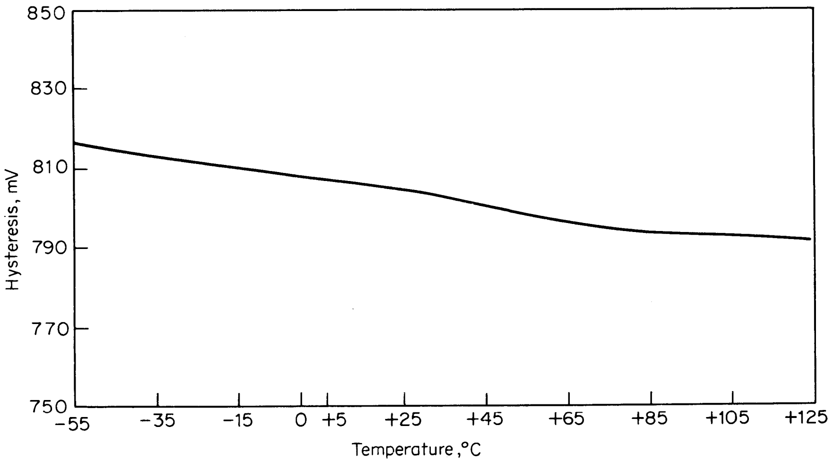


Fig. 3.45. Typical temperature variation of hysteresis, SN5413.

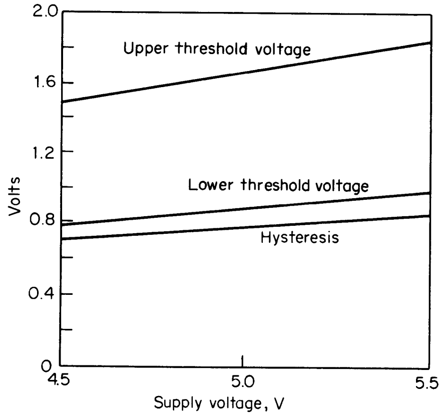


Fig. 3.46. Typical variation of thresholds and hysteresis with supply voltage, SN7413.

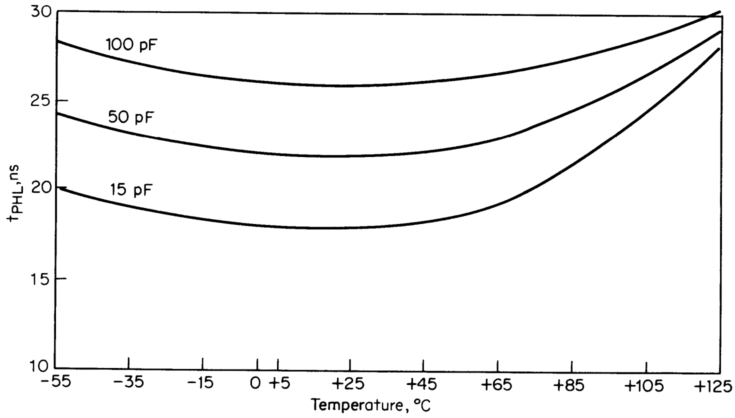


Fig. 3.47. Typical variation of t_{PLH} with temperature and load capacitance, SN5413.

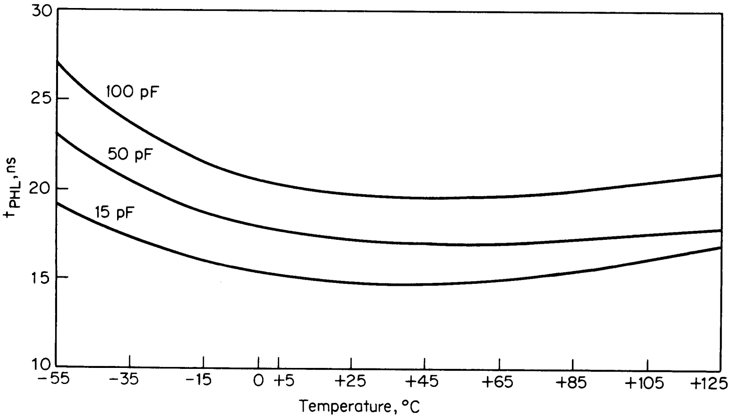


Fig. 3.48. Typical variation of t_{PHL} with temperature and load capacitance, SN5413.

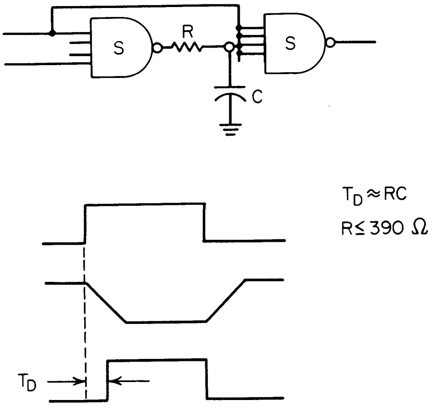

S
R
S
C
T_D
$T_D \approx RC$
$R \leq 390\ \Omega$

Figure 3.49

speed capability and the typical TTL input loads it presents, it is practical to use any spare Schmitt trigger as a standard gate.

A typical application in which the SN54/7413 is used as a delay element is illustrated by Fig. 3.49. One-half the SN54/7413 is used as a standard NAND gate.

Appendix to Chapter 3

TTL LOADING RULES

Normalized Fan-out. Fan-out capabilities shown on the individual data sheets in the TI catalog have been normalized to reflect the circuits' ability to supply a load or sink current to a number (*N*) of selected unit loads. In each of the three TTL families, the selected unit load consists of a circuit representative of the worst-case current required to drive one emitter of the multiple-emitter input transistor characteristic of that family. A summary of the values used for normalizing is shown in Table A.1. Figures A.1 and A.2 illustrate the direction of the currents.

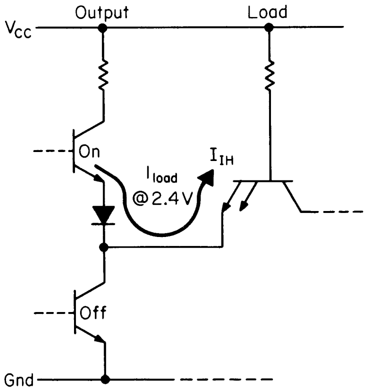

Output
Load
V_{CC}
On
Off
I_{IH}
I_{load}
@2.4V
Gnd

Fig. A.1. Logical 1 currents

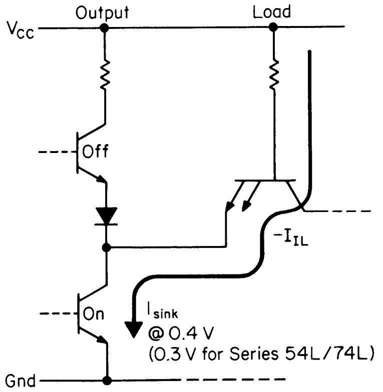

Output
Load
V_{CC}
Off
On
$-I_{IL}$
I_{sink}
@0.4 V
(0.3 V for Series 54L/74L)
Gnd

Fig. A.2. Logical 0 currents

Table A.1. Values Used for Normalizing

Series	Output state	Characteristics		Fan-out*	
		Standard totem-pole or Darlington output	Each standard input emitter	Actual	Normalized
54/74	Logical 1	$I_{load} = -400\ \mu A$ $V_{OH} = 2.4$ V min	$I_{IH} = 40\ \mu A$ max at $V_{in} = 2.4$ V	10	10
	Logical 0	$I_{sink} = 16$ mA $V_{OL} = 0.4$ V max	$I_{IL} = -1.6$ mA max at $V_{in} = 0.4$ V	10	
54H/74H	Logical 1	$I_{load} = -500\ \mu A$ $V_{OH} = 2.4$ V min	$I_{IH} = 50\ \mu A$ max at $V_{in} = 2.4$ V	10	10
	Logical 0	$I_{sink} = 20$ mA $V_{OL} = 0.4$ V max	$I_{IL} = -2$ mA max at $V_{in} = 0.4$ V	10	
54L/74L	Logical 1	$I_{load} = -100\ \mu A$ $V_{OH} = 2.4$ V min	$I_{IH} = 10\ \mu A$ max at $V_{in} = 2.4$ V	10	10†
	Logical 0	$I_{sink} = 2$ mA $V_{OL} = 0.3$ V max	$I_{IL} = -0.18$ mA max at $V_{in} = 0.3$ V	>11	

*Actual fan-out values shown here are: logical 1 fan-out = $I_{load} \div I_{IH}$
logical 0 fan-out = $I_{sink} \div I_{IL}$

†Limiting parameter is I_{load} of driving circuit.

Termination of Unused Inputs. Input current requirements for multiple emitters of the same input transistor which are connected so as to be driven by the same output are detailed in Table A.2.

This merely illustrates the fact that when unused inputs are tied to used inputs, only the $I_{in(1)}$ current requirements are increased. Most new TI TTL MSI circuits are characterized to drive 20 normalized logical 1 level loads to simplify the termination of unused inputs to used inputs.

Combining Standard, High-speed, and Low-power Circuits. Since various sections of digital logic systems have different speed requirements, a considerable improvement in system efficiency is realized when all three speed/power ranges of TI's compatible TTL are utilized. For example, Series 54H/74H high-speed circuits

Table A.2. Input Currents When Emitters Are Tied Together

Number of emitters tied together	Total input current required	
	I_{IL}	I_{IH}
1	$1 \times I_{IL}$	$1 \times I_{IH}$
2	$1 \times I_{IL}$	$2 \times I_{IH}$
3	$1 \times I_{IL}$	$3 \times I_{IH}$
⋮	⋮	⋮
N	$1 \times I_{IL}$	$N \times I_{IH}$

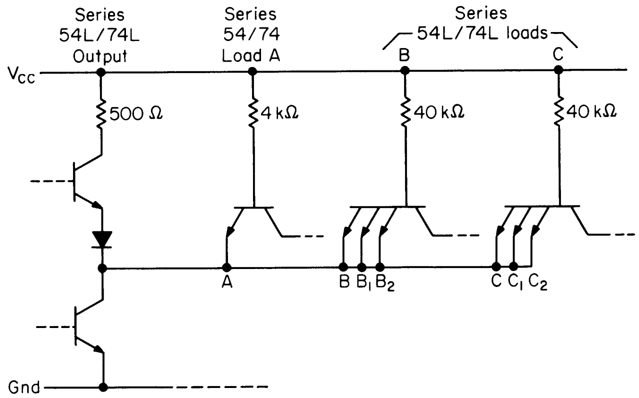


Figure A.3

may be employed in critical arithmetic sections, while standard-speed Series 54/74 and Series 54L/74L SSI and MSI circuits can be selectively used without degradation in system performance (and without special interface circuits) to minimize power consumption.

The most efficient calculations for maximum fan-out when mixing standard, high-speed, and low-power TTL circuits are achieved by using the actual current values. As an example, consider a Series 54L/74L gate (see Fig. A.3) driving one Series 54/74 load (*A*), two Series 54L/74L loads (*B* and *C*), and four unused emitters (B_1, B_2, C_1, and C_2, two on each of the 54L/74L input transistors *B* and *C*). The calculations are shown in Table A.3.

Generalized Loading Rules for the TTL Families. Attempts to normalize the three families to a single set of fan-out numbers result in the apparent loss of some drive capability when compared to the calculations using actual current values. Moreover, to be usable, the group of generalized fan-out numbers should be a small group of easily remembered values. A look at the actual normalized fan-out capability of each TTL family, when compared to input current requirements of the other two families, is provided in Table A.4.

The most significant variation between actual and generalized fan-out capabilities occurs when Series 54L/74L outputs are used to drive Series 54/74 and 54L/74L

Table A.3. Examples of Calculating Maximum Fan-out (Series 54L/74L)

Available drive		Load		
$I_{load} = 100\ \mu A$	$I_{sink} = 2$ mA	*A*	$I_{IH} = 40\ \mu A$	$I_{IL} = 1.6$ mA
		B	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
		B_1	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
		B_2	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
		C	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
		C_1	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
		C_2	$I_{IH} = 10\ \mu A$	$I_{IL} = 0.18$ mA
Less total load values: 100 μA	1.96 mA	Total	100 μA	1.96 mA
0	0.04 mA			

inputs. The generalized numbers indicate that the Series 54L/74L output will drive only one Series 54/74 load and one Series 54L/74L load simultaneously. Calculations in Table A.4 show that a Series 54L/74L output will drive at least one Series 54/74 load and two Series 54L/74L loads.

Series 54/74 input load factor	1.00
Series 54L/74L input load factor	(+)0.25
Series 54L/74L output load factor	1.25

When Series 54L/74L is used to drive 54L/74L loads, the generalized fan-out number for 54L/74L outputs is actually twice as large as when driving Series 54/74 or 54H/74H loads (see Table A.5). This can easily be compensated for by:

1. Doubling the Series 54/74 or 54H/74H input load factor and using 2.5 as fan-out capability for Series 54L/74L.

Table A.4. Actual D-C Fan-out Capabilities

Series 54/74 driving Series 54H/74H $I_{load} = 400\ \mu A \div 50\ \mu A = 8$ loads $I_{sink} = 16\ mA \div 2\ mA = 8$ loads Ⓐ	54H/74H and 54L/74L normalized to 54/74
Series 54/74 driving Series 54L/74L $I_{load} = 400\ \mu A \div 10\ \mu A = 40$* loads Ⓑ $I_{sink} = 16\ mA \div 0.18\ mA = 88.8$ loads	
Series 54H/74H driving Series 54/74 $I_{load} = 500\ \mu A \div 40\ \mu A = 12.5$ loads $I_{sink} = 20\ mA \div 1.6\ mA = 12.5$ loads Ⓒ	54/74 and 54L/74L normalized to 54H/74H
Series 54H/74H driving Series 54L/74L $I_{load} = 500\ \mu A \div 10\ \mu A = 50$* loads Ⓓ $I_{sink} = 20\ mA \div 0.18\ mA = 111.1$ loads	
Series 54L/74L driving Series 54/74 $I_{load} = 100\ \mu A \div 40\ \mu A = 2.5$ loads $I_{sink} = 2\ mA \div 1.6\ mA = 1.25$† loads Ⓔ	54/74 and 54H/74H normalized to 54L/74L
Series 54L/74L driving Series 54H/74H $I_{load} = 100\ \mu A \div 50\ \mu A = 2$ loads $I_{sink} = 2\ mA \div 2\ mA = 1$† load Ⓕ	

*Limiting parameter is I_{load} of driving circuit.
†Limiting parameter is I_{sink} of driving circuit.

Table A.5. Generalized D-C Fan-out Capabilities to Mixed Loads

Series 54/74	driving	Series 54H/74H	$10 \div 1.25 = 8$	Ⓐ	54H/74H and 54L/74L normalized to 54/74
Series 54H/74H	driving	Series 54L/74L	$10 \div 0.25 = 40$	Ⓑ	
Series 54H/74H	driving	Series 54L/74L	$12.5 \div 1 = 12.5$	Ⓒ	54/74 and 54L/74L normalized to 54H/74H
Series 54H/74H	driving	Series 54L/74L	$1.25 \div 0.25 = 50$	Ⓓ	
Series 54L/74L	driving	Series 54/74	$1.25 \div 1 = 1.25$	Ⓔ	54/74 and 54H/74H normalized to 54L/74L
Series 54/74	driving	Series 54/74	$1.25 \div 1.25 = 1$	Ⓕ	

EXAMPLE

Input load factor—Series 54/74 (one load doubled)	2.00
Input load factor—Series 54L/74L (two 0.25 loads)	(+)0.50
Output load factor—Series 54L/74L	2.50

2. Or, doubling the remaining Series 54L/74L output load factor after subtracting Series 54/74 loads.

EXAMPLE

Output load factor—Series 54L/74L	1.25
Input load factor—Series 54/74	(−)1.00
Remaining output load factor:	
Series 54L/74L	0.25
Double	$0.25 \times 2 = 0.50$
Input load factor—Series 54L/74L (two 0.25 loads)	0.50
	0.00

These fan-out rules are very useful for tabulating loading of outputs. However, if these maximums are reached for a particular output, it will be advantageous if the actual current values are considered, especially if loads remain which could be driven within the allowable maximum. A detailed listing of generalized load and fan-out factors for inputs and outputs of the three TTL families is provided in Tables A.6, A.7, A.8, and A.9.

Table A.6. Series 54/74 Generalized Loading Factors

Series 54/74			
Device	Location	Input load factor	Output load factor
SN5400/7400	Any input	1.0	
	Any output	. . .	10.0
SN5401/7401	Any input	1.0	
	Any output	. . .	10.0
SN5402/7402	Any input	1.0	
	Any output	. . .	10.0
SN5403/7403	Any input	1.0	
	Any output	. . .	10.0
SN5404/7404	Any input	1.0	
	Any output	. . .	10.0
SN5405/7405	Any input	1.0	
	Any output	. . .	10.0
SN5410/7410	Any input	1.0	
	Any output	. . .	10.0
SN5420/7420	Any input	1.0	
	Any output	. . .	10.0
SN5430/7430	Any input	1.0	
	Output	. . .	10.0
SN5440/7440	Any input	1.0	
	Any output	. . .	30.0
SN7441A	B, C, or D input	1.0	
	A input	2.0	
	Any output	. . .	N/A
SN5442/7442	Any input	1.0	
	Any output	. . .	10.0
SN5443/7443	Any input	1.0	
	Any output	. . .	10.0
SN5444/7444	Any input	1.0	
	Any output	. . .	10.0
SN5445/7445	Any input	1.0	
	Any output	. . .	12.5
SN5446/7446	Any input except BI/RBO	1.0	
	BI/RBO node	2.6	
	Any output except BI/RBO	. . .	12.5
	BI/RBO node	. . .	5.0
SN5447/7447	Any input except BI/RBO	1.0	
	BI/RBO node	2.6	
	Any output except BI/RBO	. . .	12.5
	BI/RBO node	. . .	5.0
SN5448/7448	Any input except BI/RBO	1.0	
	BI/RBO node	2.6	
	Any output except BI/RBO	. . .	4.0
	BI/RBO node	. . .	5.0
SN5449/7449	Any input	1.0	
	Any output	. . .	6.0
SN5450/7450	A, B, C, or D input	1.0	
	X and $\bar{X}$ input	N/A	
	Any output	. . .	10.0
SN5451/7451	Any input	1.0	
	Any output	. . .	10.0
SN5453/7453	A, B, C, D, E, F, G, and H input	1.0	
	X or $\bar{X}$ input	N/A	
	Output	. . .	10.0
SN5454/7454	Any input	1.0	
	Output	. . .	10.0
SN5460/7460	Any input	1.0	
	X or $\bar{X}$ output	. . .	N/A
SN5470/7470	J_1, J_2, J^*, K_1, K_2, or K^* input	1.0	
	Clock input	1.0	
	Preset or clear input	2.0	
	Q or $\bar{Q}$ output	. . .	10.0
SN5472/7472	J_1, J_2, J_3, K_1, K_2, or K_3 inputs	1.0	
	Clock input	2.0	
	Preset or clear inputs	2.0	
	Q or $\bar{Q}$ output	. . .	10.0
SN5473/7473	J or K input	1.0	
	Clock input	2.0	
	Clear input	2.0	
	Q or $\bar{Q}$ output	. . .	10.0
SN5474/7474	D input	1.0	
	Clock input	2.0	
	Preset input	2.0	
	Clear input	3.0	
	Q or $\bar{Q}$ output	. . .	10.0

Table A.7. Series 54/74 Generalized Loading Factors

Series 54/74			
Device	Location	Input load factor	Output load factor
SN5475/7475	D_1, D_2, D_3, or D_4 input	2.0	
	Clock 1-2, or clock 3-4 input	4.0	
	Any output	...	10.0
SN5476/7476	J or K input	1.0	
	Clock input	2.0	
	Clear input	2.0	
	Preset input	2.0	
	Q or $\bar{Q}$ output	...	10.0
SN5477/7477	Any D input	2.0	
	Clock 1-2 or clock 3-4 input	4.0	
	Any output	...	10.0
SN5480/7480	A_1, A_2, B_1, B_2, A_C, or B_C input	1.0	
	A^* or B^* input	1.65	
	C_n input	5.0	
	Σ or $\bar{\Sigma}$ output	...	10.0
	$\overline{C_{n+1}}$ output	...	5.0
	A^* or B^* output	...	3.0
SN5481/7481	Write 1 and write 0 input	1.0	
	X and Y inputs:		
	Logical 0	7.0	
	Logical 1	10.0	
	S_0 or S_1: SN5481	...	12.5
	SN7481	...	25.0
SN5482/7482	A_1 or B_1 input	4.0	
	A_2 or B_2 input	1.0	
	C_0 input	4.0	
	C_2 output	...	5.0
	Σ_1 or Σ_2 output	...	10.0
SN5483/7483	A_1, B_1, A_3, or B_3 input	4.0	
	A_2, B_2, A_4, or B_4 input	1.0	
	C_0 input	4.0	
	C_4 output	...	5.0
	Σ_1, Σ_2, Σ_3, or Σ_4 output	...	10.0
SN5484/7484	Write 1 and write 0 input	1.0	
	X and Y inputs:		
	Logical 0	7.0	
	Logical 1	10.0	
	S_0 or S_1	...	25.0
SN5486/7486	Any input	1.0	
	Any output:		
	Logical 1	...	20.0
	Logical 0	...	10.0
SN5490/7490	$R_{0(1)}$, $R_{0(2)}$, $R_{9(1)}$, or $R_{9(2)}$ input	1.0	
	BD input	4.0	
	A input	2.0	
	Any output	...	10.0
SN5491A/7491A	A or B input	1.0	
	$\overline{CP}$ input	1.0	
	Q or $\bar{Q}$ output	...	10.0
SN5492/7492	$R_{0(1)}$ or $R_{0(2)}$ input	1.0	
	BC input	4.0	
	A input	2.0	
	Any output	...	10.0
SN5493/7493	$R_{0(1)}$ or $R_{0(2)}$ input	1.0	
	B input	2.0	
	A input	2.0	
	Any output	...	10.0
SN5494/7494	Any input except preset 1 or 2	1.0	
	Preset 1 or preset 2 input	4.0	
	Any output	...	10.0
SN5495/7495	Any input except mode control	1.0	
	Mode control input	2.0	
	Any output	...	10.0
SN5496/7496	Any input except preset	1.0	
	Preset input	5.0	
	Any output	...	10.0
SN54100/74100	Any D input	2.0	
	Clock 1 or clock 2 input	8.0	
	Any output	...	10.0
SN54107/74107	J or K input	1.0	
	Clock input	2.0	
	Clear input	2.0	
	Q or $\bar{Q}$ output	...	10.0
SN54121/74121	A_1 and A_2 inputs	1.0	
	B input	2.0	
	Any output	...	10.0
SN54145/74145	Any input	1.0	
	Any output	...	12.5
SN54150/74150	Any input	1.0	
	Any output:		
	Logical 1	...	20.0
	Logical 0	...	10.0
SN54151/74151	Any input	1.0	
	Any output:		
	Logical 1	...	20.0
	Logical 0	...	10.0
SN54152/74152	Any input	1.0	
	Any output:		
	Logical 1	...	20.0
	Logical 0	...	10.0
SN54180/74180	Data input	1.0	
	Odd or even inputs	2.0	
	Any output:		
	Logical 1	...	20.0
	Logical 0	...	10.0

Table A.8. Series 54H/74H Generalized Loading Factors

Series 54H/74H			
Device	Location	Input load factor	Output load factor
SN54H00/74H00	Any input	1.25	
	Any output	....	12.5
SN54H01/74H01	Any input	1.25	
	Any output	....	12.5
SN54H04/74H04	Any input	1.25	
	Any output	....	12.5
SN54H05/74H05	Any input	1.25	
	Any output	....	12.5
SN54H10/74H10	Any input	1.25	
	Any output	....	12.5
SN54H11/74H11	Any input	1.25	
	Any output	....	12.5
SN54H20/74H20	Any input	1.25	
	Any output	....	12.5
SN54H21/74H21	Any input	1.25	
	Any output	....	12.5
SN54H22/74H22	Any input	1.25	
	Any output	....	12.5
SN54H30/74H30	Any input	1.25	
	Any output	....	12.5
SN54H40/74H40	Any input	2.5	
	Any output	....	37.5
SN54H50/74H50	*A*, *B*, *C*, or *D* input	1.25	
	X or $\bar{X}$ input	N/A	
	Any output	....	12.5
SN54H51/74H51	Any input	1.25	
	Any output	....	12.5
SN54H52/74H52	*A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, or *J* input	1.25	
	X input	N/A	
	Output	....	12.5
SN54H53/74H53	*A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, or *J* input	1.25	
	X or $\bar{X}$ input	N/A	
	Output	....	12.5
SN54H54/74H54	Any input	1.25	
	Output	....	12.5
SN54H55/74H55	*A*, *B*, *C*, *D*, *E*, *F*, *G*, or *H* input	1.25	
	X or $\bar{X}$ input	N/A	
	Output	....	12.5
SN54H60/74H60	Any input	1.25	
	X or $\bar{X}$ output	....	N/A
SN54H61/74H61	Any input	1.25	
	Any output	....	N/A

Series 54H/74H			
Device	Location	Input load factor	Output load factor
SN54H62/74H62	Any input	1.25	
	X or $\bar{X}$ output	....	N/A
SN54H71/74H71	*J1A*, *J2A*, *J1B*, *J2B*, *K1A*, *K2A*, *K1B*, or *K2B* input	1.25	
	Clock input	2.50	
	Preset input	3.75	
	Q or $\bar{Q}$ output	....	12.5
SN54H72/74H72	*J*1, *J*2, *J*3, *K*1, *K*2, or *K*3 input	1.25	
	Preset or clear input	2.50	
	Clock input	1.25	
	Q or $\bar{Q}$ output	....	12.5
SN54H73/74H73	*J*, *K*, or clock input	1.25	
	Clear input	2.50	
	Q or $\bar{Q}$ output	....	12.5
SN54H76/74H76	*J*, *K*, or clock input	1.25	
	Clear or preset input	2.50	
	Q or $\bar{Q}$ output	....	12.5
SN54H78/74H78	*J* or *K* input	1.25	
	Clear input	5.00	
	Clock or preset input	2.50	
	Q or $\bar{Q}$ output	....	12.5
SN54H87/74H87	Any input	1.25	
	Any output	....	12.5
SN54H101/74H101	*J* or *K* input	1.25	
	Preset input	2.50	
	Clock input	3.0	
	Any output	....	12.5
SN54H102/74H102	*J* or *K* input	1.25	
	Preset or clear input	2.50	
	Clock input	3.0	
	Any output	....	12.5
SN54H103/74H103	*J* or *K* input	1.25	
	Clear input	2.50	
	Clock input	3.0	
	Any output	....	12.5
SN54H106/74H106	*J* or *K* input	1.25	
	Preset or clear input	2.5	
	Clock input	3.0	
	Any output	....	12.5
SN54H108/74H108	*J* or *K* input	1.25	
	Preset input	2.5	
	Clear input	5.0	
	Clock input	6.0	
	Any output	....	12.5

Table A.9. Series 54L/74L Generalized Loading Factors

Series 54L/74L				
Device	Location	Input load factor	Output load factor	Output load factor*
SN54L00/74L00	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L04/74L04	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L10/74L10	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L20/74L20	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L30/74L30	Any input	0.25		
	Output	. . . .	1.25	2.5*
SN54L51/74L51	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L54/74L54	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L55/74L55	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L71/74L71	$R1$, $R2$, $R3$, $S1$, $S2$, or $S3$ input	0.25		
	Clock input	0.50		
	Preset or clear input	0.50		
	Q or $\bar{Q}$ output	. . . .	1.25	2.5*
SN54L72/74L72	$J1$, $J2$, $J3$, $K1$, $K2$, or $K3$ input	0.25		
	Clock input	0.50		
	Preset or clear input	0.50		
	Q or $\bar{Q}$ output	. . . .	1.25	2.5*
SN54L73/74L73	J or K input	0.25		
	Clock input	0.50		
	Clear input	0.50		
	Q or $\bar{Q}$ output	. . . .	1.25	2.5*
SN54L78/74L78	J or K input	0.25		
	Preset input	0.50		
	Clear or clock input	1.00		
	Q or $\bar{Q}$ output	. . . .	1.25	2.5*
SN54L86/74L86	Any input	0.50		
	Any output	. . . .	1.25	2.5*
SN54L91/74L91	Any input	0.25		
	Any output	. . . .	1.25	2.5*
SN54L93/74L93	$R_{0(1)}$ or $R_{0(2)}$ input	0.25		
	B input	0.50		
	A input	0.50		
	Any output	. . . .	1.25	2.5*
SN54L95/74L95	Any input except mode control	0.25		
	Mode control input	0.50		
	Any output	. . . .	1.25	2.5*

*Use these numbers when driving Series 54L/74L loads only.

4

Extended-range Operation

Although data sheet specifications set limits of operation for TTL devices beyond which no guarantees are made, it is frequently desirable to operate outside these limits. Such an approach can be justified when the specifications are based on worst-case testing of all parameters. It is sometimes acceptable, for instance, to risk losing half the 400-mV d-c safety margin in order to increase the fan-out. (Even though loading may be increased, in most cases a safety margin exceeding the data sheet guarantee would still exist.) It is the purpose of this section to show the capability of the devices and their variations.

It must be understood that (except in the case of a specific contract between vendor and customer) a component manufacturer cannot accept responsibility for the performance of a device that is so operated that catalog data specifications are exceeded.

4.1 INTEGRATED-CIRCUIT COMPONENTS

The three components basic to a monolithic circuit—transistors, diodes, and resistors—determine the capabilities of the circuit. Hence the characteristics and reactions of these components to external factors must be understood before we attempt a description of the circuit performance under variable influences. Secondary factors such as capacitance, although certainly not constant, are not important, and they can safely be ignored for the purpose at hand.

Resistance. The sheet resistance of integrated-circuit material used to fabricate standard TTL resistors is 130 Ω per square and increases with temperature at about 1 percent per °C above 0°C. Below 0°C, the temperature coefficient of resistance can be considered 0. On the other hand, the high resistances required by low-power TTL are provided by a different doping level, and the design center is 260 Ω per square. Figure 4.1 shows the typical variation of resistance with temperature.

Other resistances, such as the R_{CS} of transistors, diode forward resistance, and substrate resistance, also vary directly with temperature. While not very precise for these resistances, the variations shown in Fig. 4.1 may be applied with acceptable accuracy.

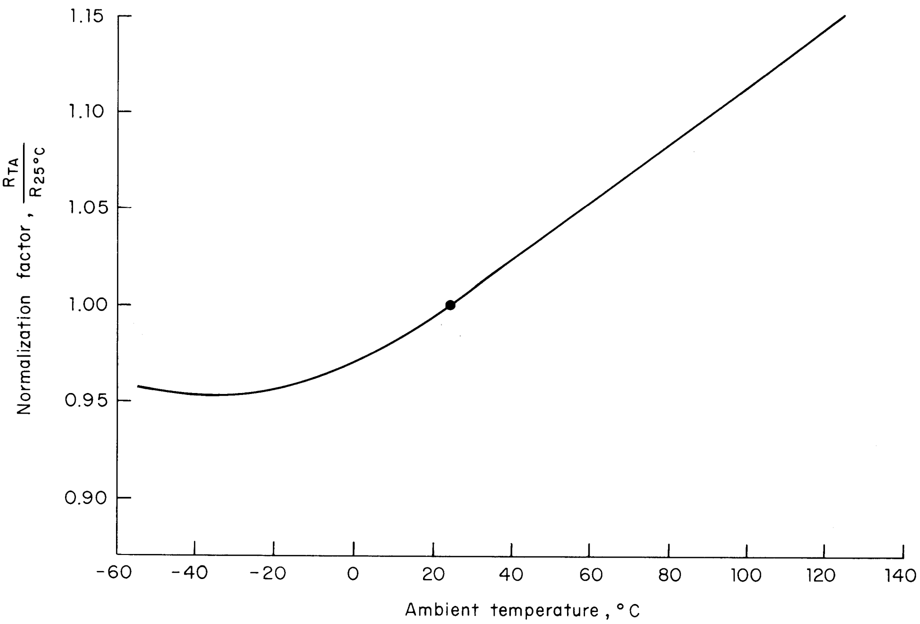


Fig. 4.1. Typical integrated-circuit resistance versus temperature.

Diodes. Diode characteristics of interest are the forward resistance, the forward conductor voltage V_F, and sometimes the reverse breakdown voltage.

Reverse leakage, while worthy of investigation, is a difficult factor to analyze. The normal integrated-circuit diode is a transistor with a collector-base short circuit, and the highly temperature-dependent reverse leakage is unimportant in contrast with the apparent leakage caused by the inverse h_{FE} of the transistor. These leakages

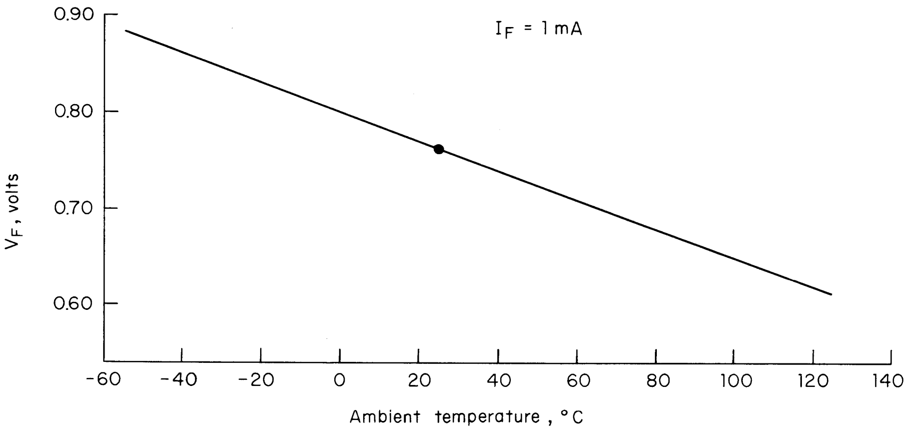


Fig. 4.2. Typical V_F of integrated-circuit diodes and base-emitter junctions versus temperature.

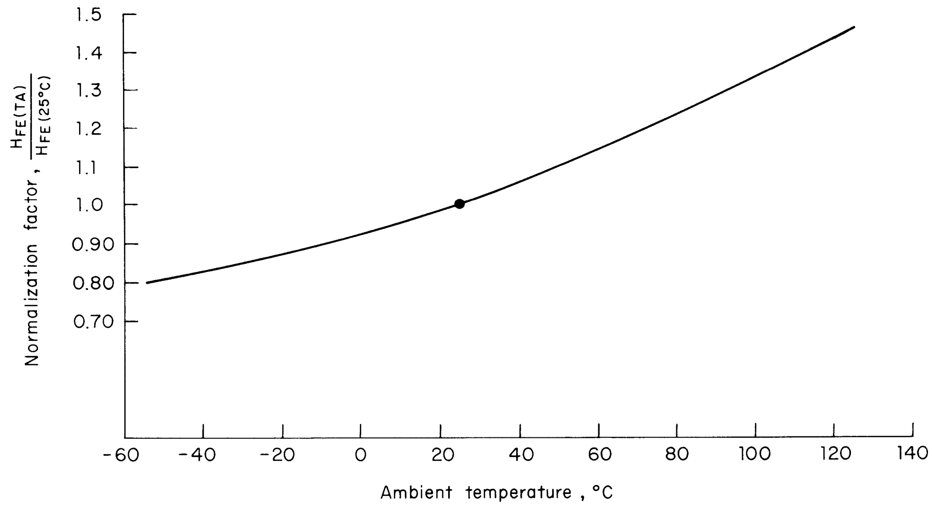


Fig. 4.3. Typical integrated-circuit transistor h_{FE} versus temperature.

are found to increase by a factor of 2 between 25 and 100°C, and they change very little below 25°C.

The typical emitter-base junction forward voltage V_F of 0.73 V at 1 mA consists of typically 0.6 V offset, plus an IR drop of 0.13 V. Figure 4.2 indicates the typical V_F and its variation with temperature.

When reverse-biased, the emitter-base junction goes into breakdown conduction somewhere close to 6 V. Where collector-base junctions or collector-substrate junctions must be considered, a wide range of breakdown voltages must be allowed. These values are generally higher than emitter-base junction breakdown voltages and range from less than 10 V to greater than 50 V.

Transistors. The breakdown voltages, forward emitter-base voltage, bulk resistance, and leakages already mentioned apply also to the active transistor. The major remaining parameter, and one of the most important, is the h_{FE}. An increase in temperature results in a proportional increase in current gain. The shift of h_{FE} with temperature is indicated in Fig. 4.3. This plot is useful for predicting device parameter shift, although precise figures are difficult to obtain because of compensating and additive factors, such as resistor value, drift, and V_{BE} shift.

4.2 DEVICE REACTION TO EXTERNAL INFLUENCES

With the internal component parameter variations as a base, it is possible to predict in general terms the way the total device will react. Actual data are used as basis for extended-range design.

Output Stage. The logical 1 output voltage is set by the V_{CC} supply and the various voltage drops in the path to the output (see Fig. 4.4). At low currents, there is one diode drop (V_F), one V_{BE} drop, and the IR drop across the base resistor. At higher currents, the transistor saturates and the emitter-follower action is lost.

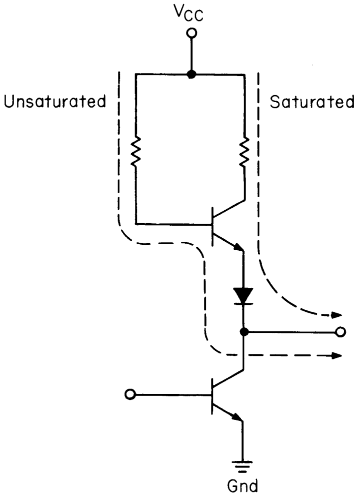


Fig. 4.4. TTL output circuit showing accumulation of voltage drops for a logical 1 output.

In this condition, the voltage drops are the V_F plus the IR drops of the transistor R_{CS} and the collector resistor. The output voltage follows changes in V_{CC} almost volt for volt. Testing is done at worst-case low V_{CC}, and if higher values are used, the difference can be expected to add to the guaranteed data sheet value.

Since resistor values go up with temperature and diode V_F goes down, these changes tend to affect each other. In the unsaturated state, the diode and V_{BE} changes dominate, and the output voltage can be expected to rise with temperature. When saturated, there is more resistive effect, but still there is a slight tendency for the voltage to increase with temperature. The low impedances involved reduce the effects of leakage to negligible levels.

Similarly the lowest allowable supply voltage is used when testing a logical 0 output, since the lower voltage reduces the base drive to the output transistor and makes it more difficult to keep the transistor in saturation. Figure 4.5 shows the source paths for the output base current. The junction characteristics again override

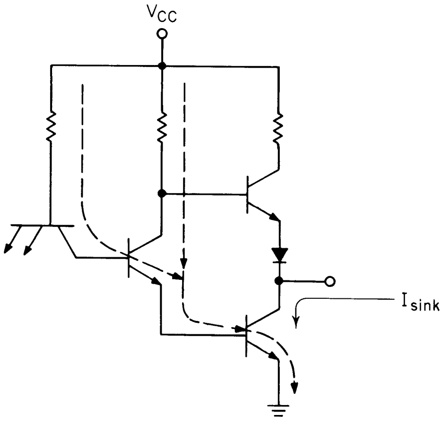


Fig. 4.5. TTL circuit showing the paths for base current to the sinking transistor.

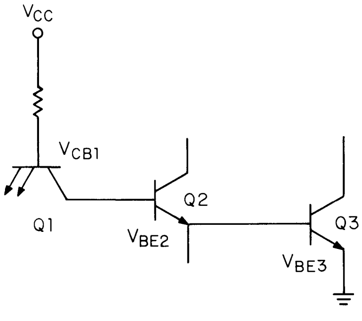


Fig. 4.6. Accumulation of voltage drops related to switching threshold.

the resistive change, and the base drive decreases directly with temperature, especially below 0°C, where the resistor change rate decreases.

The R_{CS} of the output transistor is lower at low temperature, resulting in a slightly lower logical 0 level with standard fan-out. The higher current-handling ability is seriously curtailed by loss of saturation caused by the decreased base drive, and complicated by the decreased h_{FE} of the transistors. Data show the change in output capability to be essentially linear with temperature at about 0.15 mA per °C. Similarly, the effects of V_{CC} appear to be linear at about 8 to 10 mA per V.

Input Stage. The input threshold *Vth* as seen in Fig. 4.6 is set by the accumulation of $V_{BE(Q2)} + V_{BE(Q3)} + V_{CB(Q1)} - V_{BE(Q1)}$. Thus it varies predictably with temperature at twice the temperature coefficient of a single silicon junction. Data show it decreasing with rising temperature at 4.2 mV per °C. This factor is of interest primarily when driving from other than normal logic sources (including overloaded logic devices).

Above the threshold, the input emitter-base junction is reverse-biased, and only leakage and inverse h_{FE} current flows; this current is typically so small that even a load of 10 inputs combined rarely exceeds the guarantee for a single input and deserves little further consideration unless the input is to be driven more positive than the 5.5-V data sheet limit. Avalanche breakdown occurs if much over 6-V differential between the emitter and any other emitter or the collector is allowed. Operation in the breakdown region can be tolerated if the current is limited to 2 mA or less for the SN54/74 and SN54H/74H, or to 0.5 mA for the SN54L/74L.

When an input is driven below the threshold, the current switches from the few microamps leakage into the device to about 0.8 mA out of the device, and then increases linearly to a guaranteed maximum at a specified V_{IL} of 0.40 V. When using the typical curves as a design basis, it should also be noted (Fig. 4.7) that as the voltage V_{IL} at the input increases, the current decreases:

$$I_{IL} = \frac{V_{CC} - V_{BE(Q1)} - V_{IL}}{R_{B(Q1)}}$$

If an input is driven below 0 V, current increases linearly until the input clamping diode becomes forward-biased (see Figs. 4.7 and 4.12). At this point current increases rapidly, and it must be limited to less than 20 mA for the SN54/74 and SN54H/74H devices, or to 2 mA for the SN54L/74L.

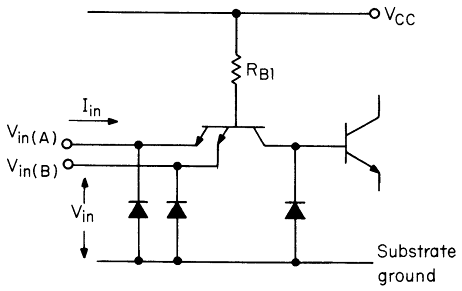


Fig. 4.7. Circuit for measurement of TTL gate input current versus input voltage.

Output Drive. Output 0-state drive (sinking mode) capability is given in Fig. 4.8 from −55 to +125°C for three values of V_{CC} (normalized to a V_{CC} of 5.0 V at 25°C). As an example, when V_{CC} = 5.0 V at −55°C, the sink capability of the SN5400 is down to approximately 40 percent of the 25°C and V_{CC} = 4.5 V value. Now if the SN5400, which is guaranteed for 16 mA at 4.5 V and −55°C, is used at 5.0 V at 20°C, it should be able to sink approximately 280 percent of 16 mA, or 45 mA. Similar extrapolation may be made to any other point for the device.

The solid curve is for the standard V_{OL} = 0.40 V (0.3 V for SN54L/74L) maximum output. Where noise margin is not critical, the dashed curve, taken at a maximum logical 0 level of 0.80 V (0.70 V for SN54L/74L), may be used, again normalized to the same point as the V_{OL} = 0.40 V curves. For instance, if V_{OL} = 0.80 V is acceptable, and the V_{CC} is 5.5 V, then approximately 300 percent of 16 mA, or 48.0 mA, is available at −55°C instead of the guaranteed 16 mA. A typical value for V_{OL} = 0.60 V can be found midway between the 0.40 and the 0.80 values. For the Series SN74 devices, which are specified only to 0°C, a similar readoff is possible

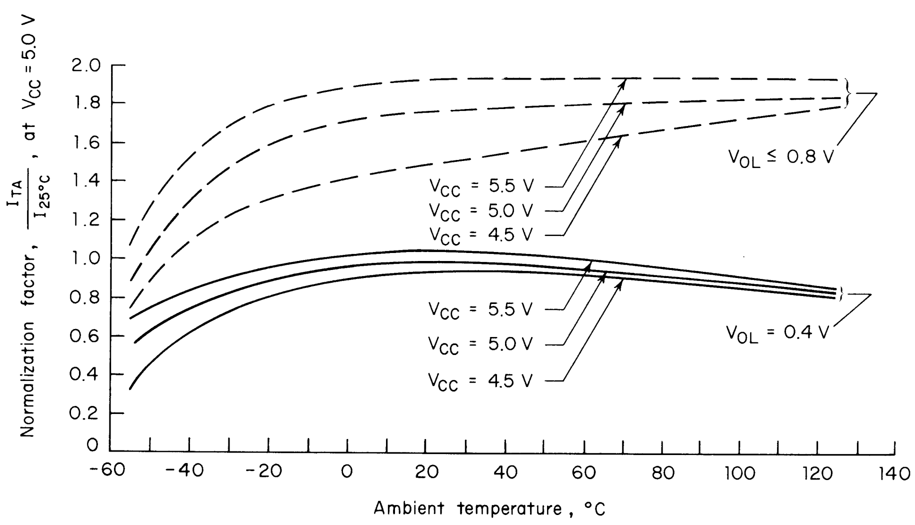


Fig. 4.8. Output sink-current capability (logical 0): (*a*) 54/74.

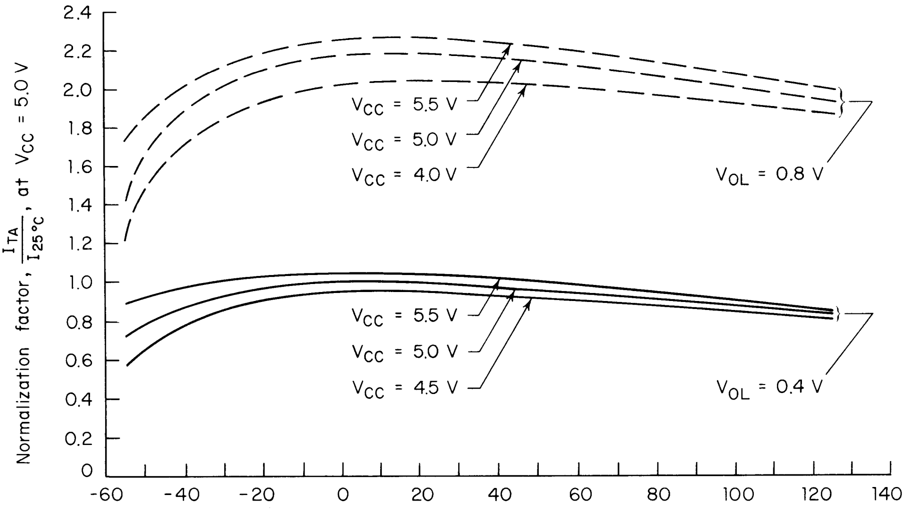


Fig. 4.8. (*b*) 54/74H.

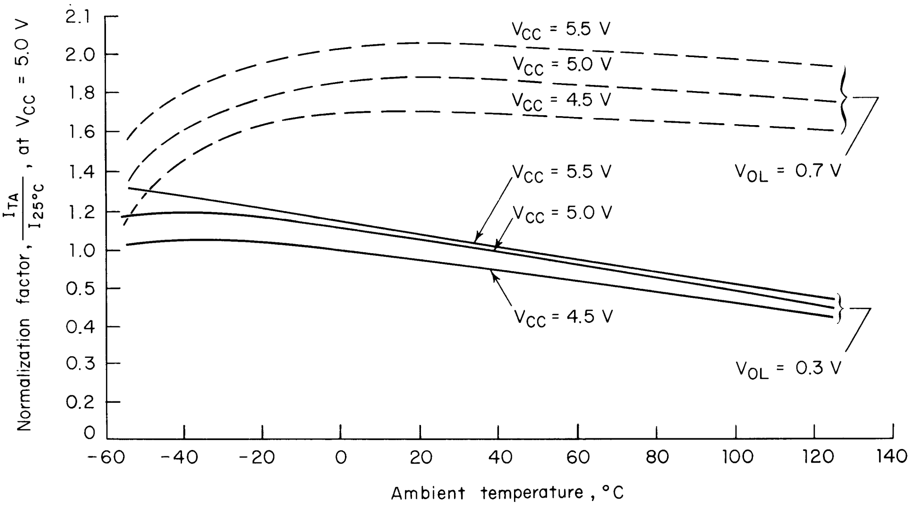


Fig. 4.8. (*c*) 54/74L.

for use at lower temperatures. Use at higher than the rated temperature is not recommended, since leakage is usually the culprit in loss of capability, and its effect is difficult to characterize.

A similar presentation is made for the output logical 1 state at $V_{OH} = 2.4$ V and 2.0 V. Percentage changes are read off in exactly the same way as shown in Fig. 4.9.

In addition, 0-state and 1-state V_0/I_0 curves are given in Figs. 4.10 and 4.11. Principally, these are for use at very low 0-state currents and very high 1-state currents and for indicating typical values when the guarantees are exceeded. When interfacing with other devices, the gate must frequently drive high currents while sinking very little. Note the very low 0-state output voltage at low currents.

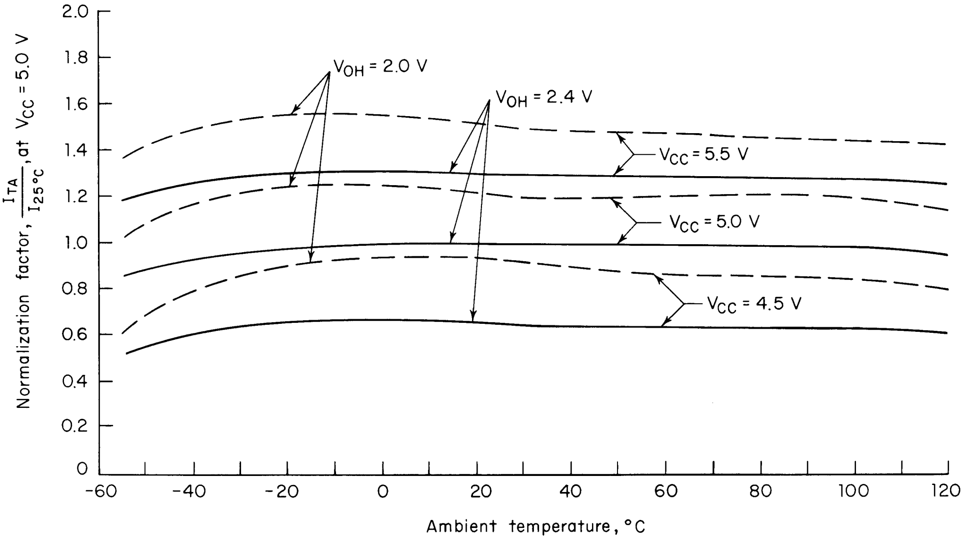


Fig. 4.9. Output source current capability (logical 1): (*a*) 54/74.

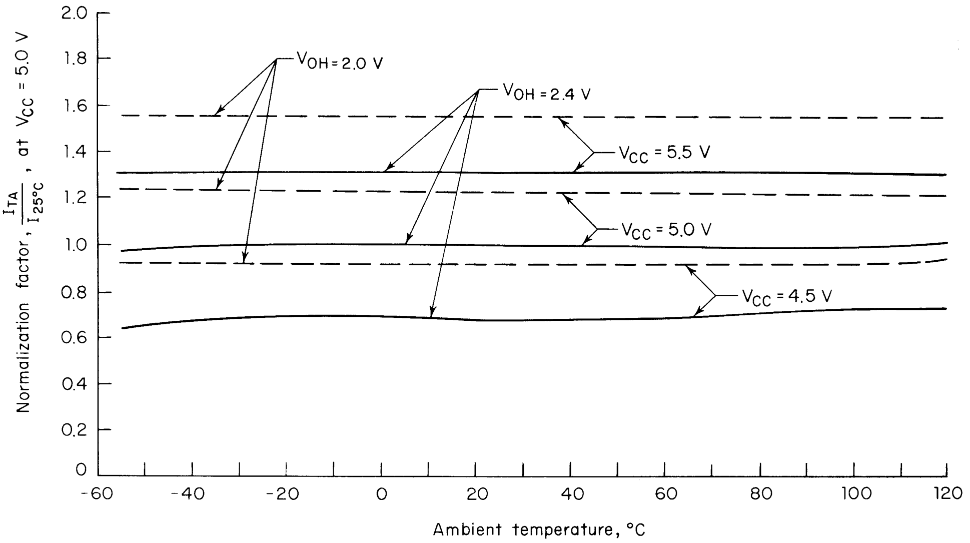


Fig. 4.9. (*b*) 54/74H.

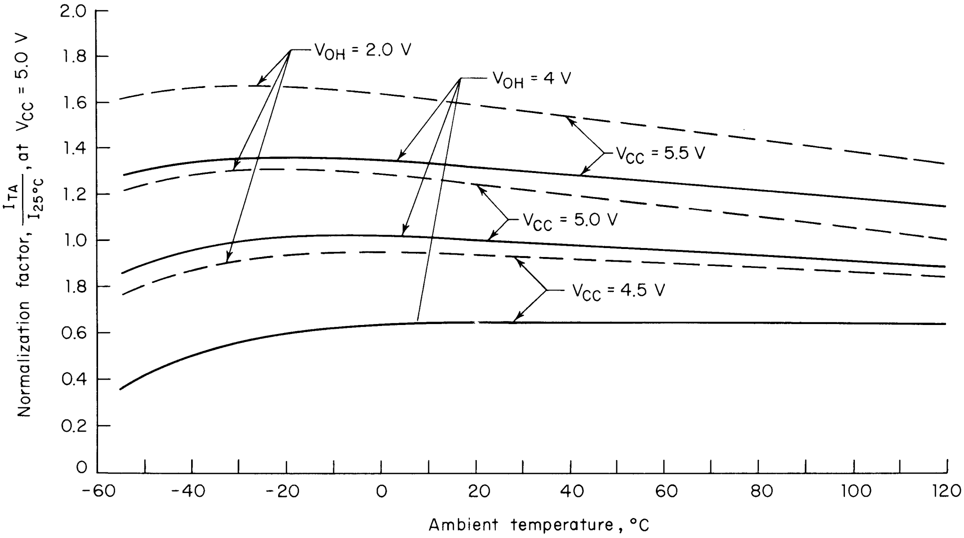


Fig. 4.9. (*c*) 54/74L.

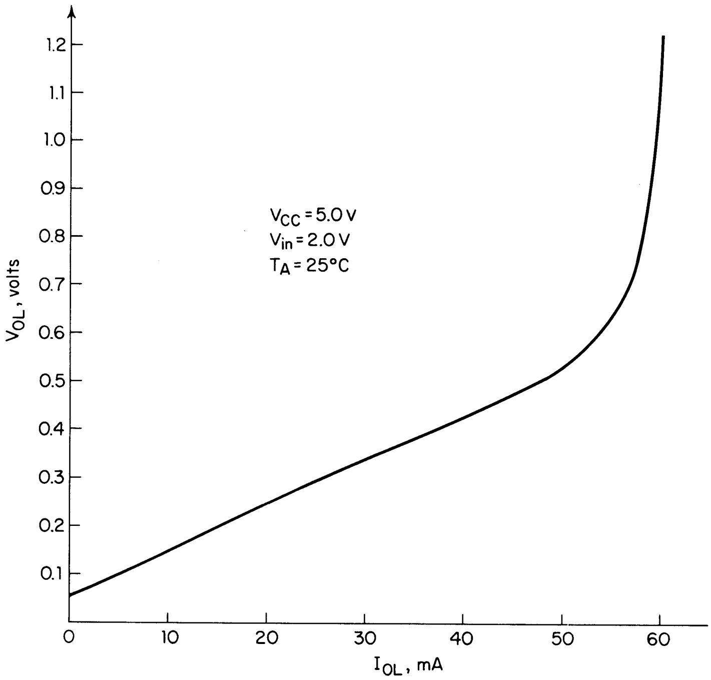


Fig. 4.10. V_{OL} *vs.* I_{OL}: (*a*) SN7400.

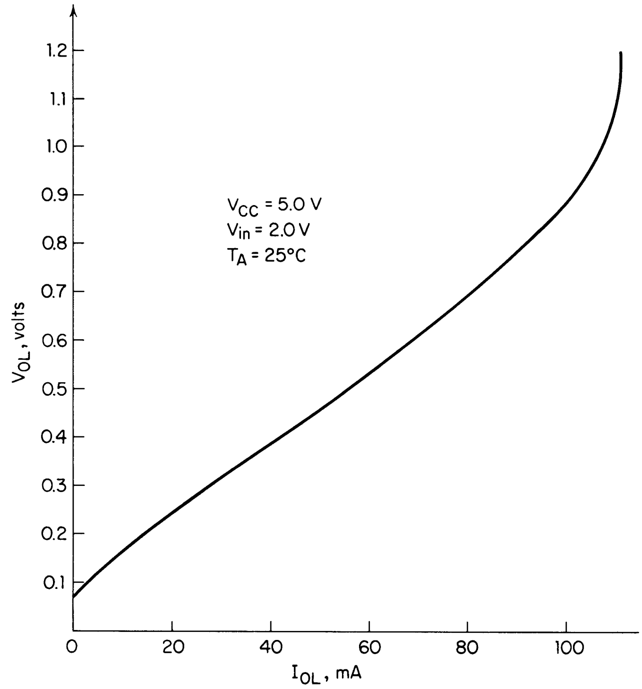


Fig. 4.10. (*b*) SN74H00N.

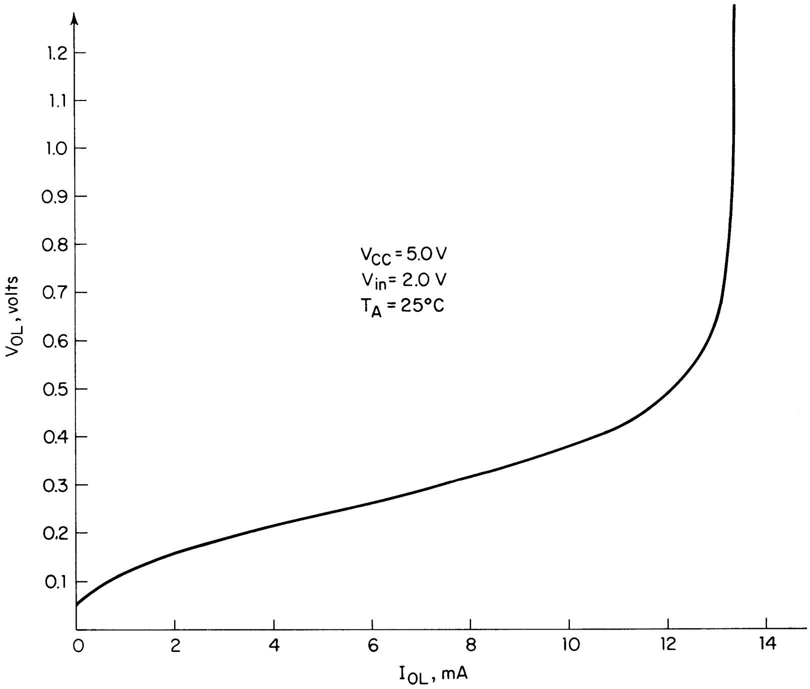


Fig. 4.10. (*c*) SN74L00.

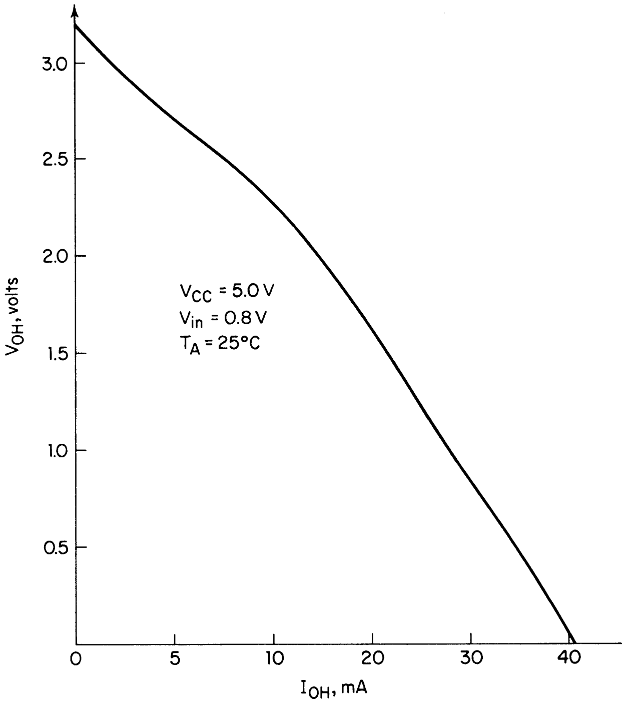


Fig. 4.11. V_{OL} *vs.* I_{OH}: (*a*) SN7400N.

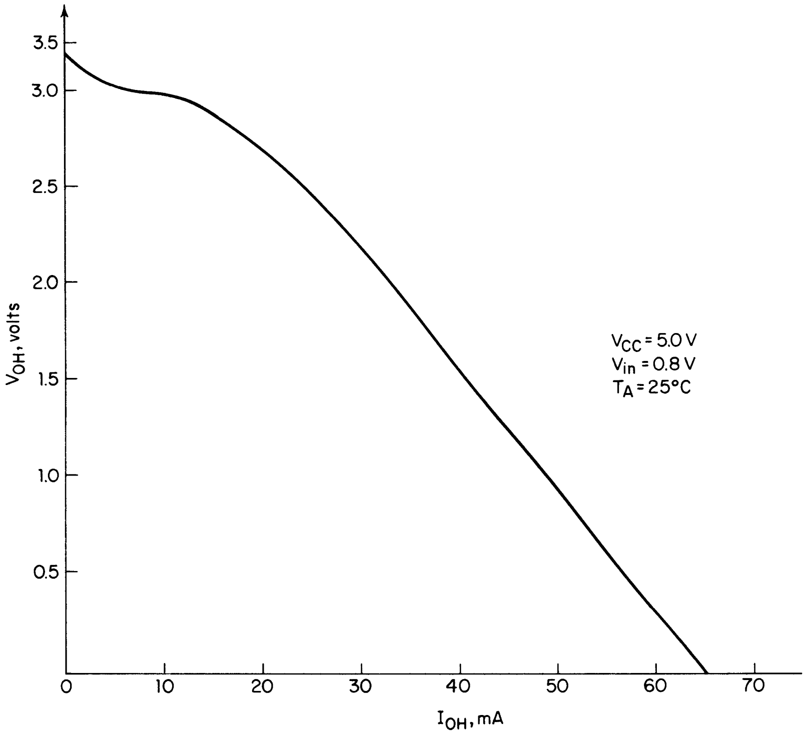


Fig. 4.11. (*b*) SN74H00N.

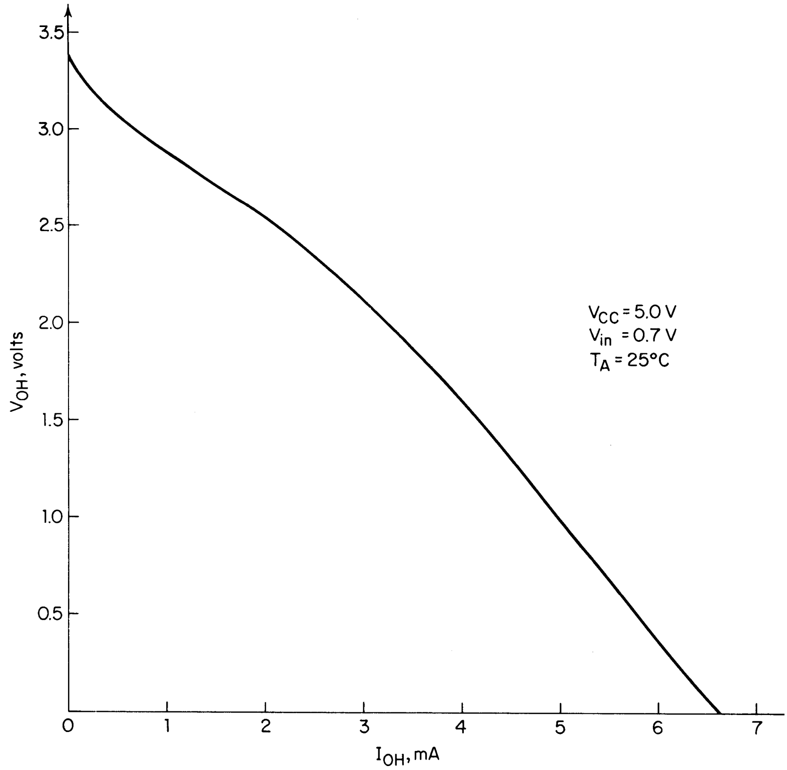


Fig. 4.11. (*c*) SN74L00N.

Input Loading. Input current varies only slightly with temperature. What is more important is the variation with V_{CC} and input voltage. The typical curves for input current with varying V_I are given in Fig. 4.12. Worst case is given for highest V_{CC} on the data sheet. The full voltage range curve shows the breakdown points, positive and negative, and shows also what happens in the vicinity of the threshold as V_I is varied. A combination of the input curves must be used at times. For example, if it is desired to operate at $V_{IL} = 0.80$ V at maximum possible fan-out for the driving gates, worst-case driving current can be obtained from Fig. 4.8, and the typical driving current from Fig. 4.10, plus data sheets of course. The input current would be decreased by about 20 percent in going from a V_{IL} of 0.4 V to 0.8 V; this is from Fig. 4.12*a*. The typical logical 0 fan-out for devices using these modified noise-margin parameters would be calculated:

$$N_L = \frac{I'_O}{I'_I}$$

where $I'_O = I_{OL}$ at $V_{OL} = 0.8$ V
$I'_I = I_{IL}$ at $V_{IL} = 0.8$ V
N_L = fan-out for logical 0 output

For example, using the standard Series 54/74 (Figs. 4.10*a* and 4.12*a*),

$$N_L = \frac{58 \text{ mA}}{0.8 \text{ mA}} = 72$$

4.3 VOLTAGE BREAKDOWN FOR INPUT AND OUTPUT

Operation in the breakdown region is not prohibited so long as power dissipation is limited and the possible device instability is not a problem. This instability is due to negative resistances in the snap-back characteristic which usually exists during breakdown.

Input breakdowns, as illustrated in Fig. 4.12, occur when driving from high-voltage sources. As previously recommended, current must be limited by source resistance or added series resistance.

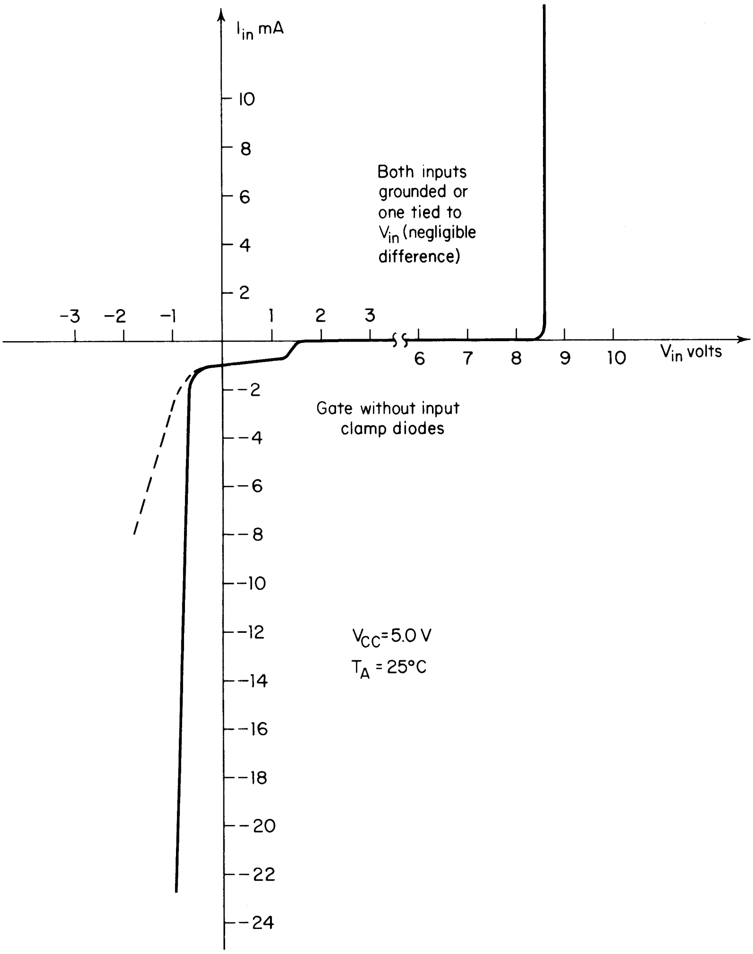


Fig. 4.12. Typical I_{in} *vs.* V_{in} for a gate input: (*a*) SN7400.

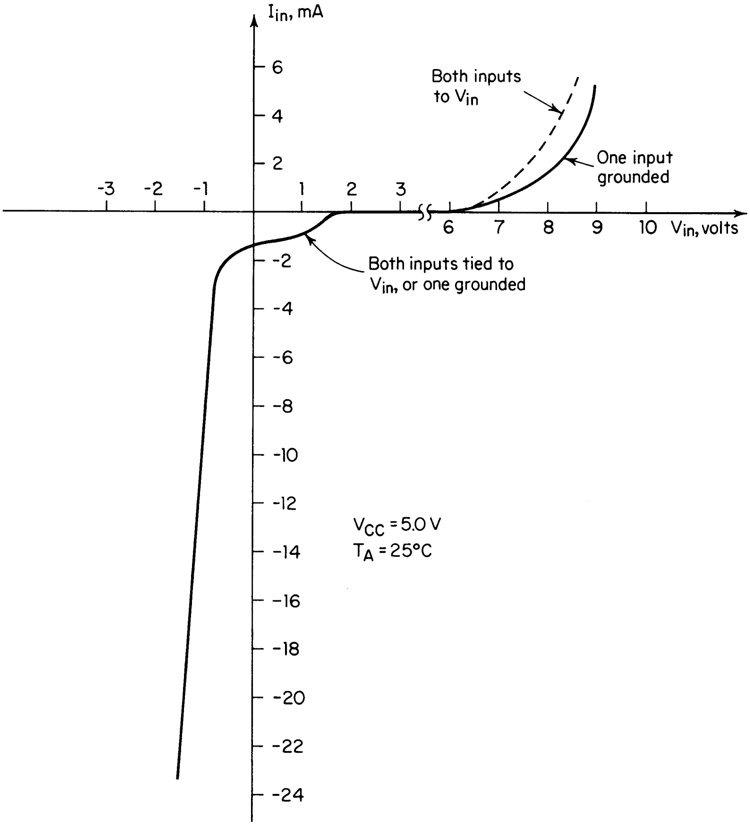


Fig. 4.12. (*b*) SN7400.

Table 4.1. Thermal Impedances for Various Package Types*

Package		Thermal impedance, °C/W	
Description	Suffix	$\theta_{JC} \pm 5\%$†	$\theta_{JX} \pm 15\%$‡
8-pin plastic	P	52	95
14-pin plastic	N	40	80
16-pin plastic	N	36	67
24-pin plastic	N	37	62
14-pin C-DIP	J	27	80
16-pin C-DIP	J	27 (est.)	80 (est.)
14-pin flat (alloy)	S	70	109
14-pin flat (frit)	F	123	172
14-pin C-Pack	. .	27	
10-pin TO-5 (alloy)	L	39	125

*The values given are nominal and are not guaranteed. Test bars were used in measurements for uniformity. Measurements were made with 300-mW dissipation.

†θ_{JC} = thermal impedance from junction to case using freon as a heat sink.

‡θ_{JX} = thermal impedance from junction to still air ambient with package in a socket.

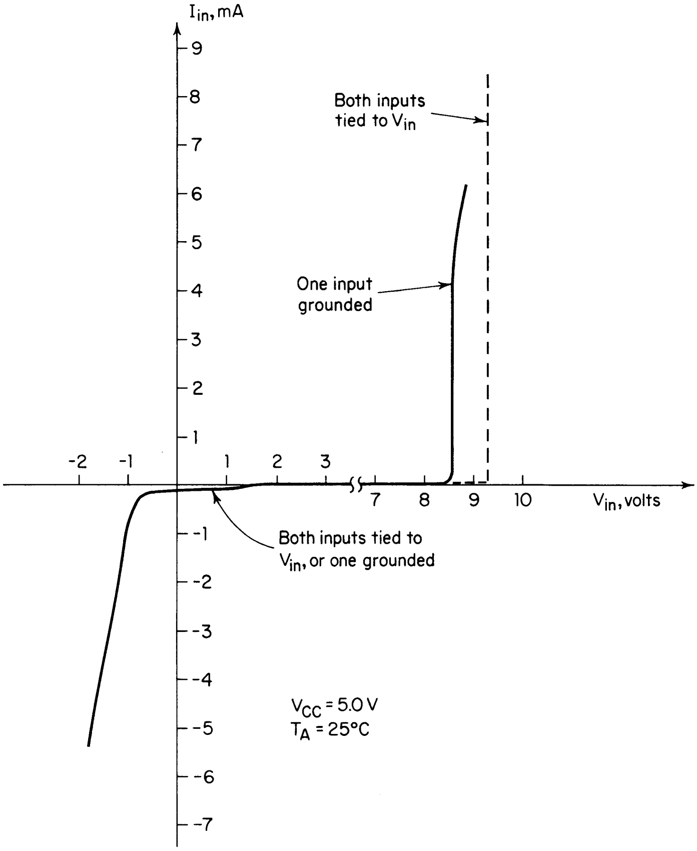


Fig. 4.12. (*c*) SN74L00.

Output breakdown is not specified for the active pull-up output. It is safe to apply positive voltages of less than 8 V in the high output state. Voltages greater than 8 V must be accompanied by resistance, to limit device power dissipation.

Open-collector device voltage ratings should be observed. When they must be exceeded, sufficient resistance should be inserted to limit dissipation, considering the snap-back characteristic of the output breakdown.

Under any conditions, maximum specified dissipation of an IC package should not be exceeded. Table 4.1 lists the typical values for thermal impedances of the packages presently available. Power may be primarily dissipated in a single gate, except for the 54L/74L devices, or it may be distributed randomly. For the SN54L/74L devices, breakdown voltage limits apply prior to the power dissipation limit $P_{\max}$. For example, a 30-V supply (V_{BB}) is to be connected to a device that is rated at 30 V but snaps back to a voltage V_{S-B} of 15 V, which is essentially $V_{(BR)CEO}$.

A safe value of series resistance R_S can be found:

$$R_S = \frac{V_{BB} - V_{S-B}}{P_{\max}/V_{S-B}} = \frac{(V_{BB} - V_{S-B})V_{S-B}}{P_{\max}}$$

$$= \frac{(30 - 15)15}{0.25} = 900\ \Omega$$

The user of devices in this mode should be cautioned that oscillation frequently does occur. Although it is usually not harmful in the immediate area, it does couple into other logic, and this eventuality must be considered.

5

Noise Considerations

Few subjects are so inclusive in scope and varied in connotation as the generalized term *noise*. The specific area of interest here is extraneous voltages and currents in digital logic systems.

Figure 5.1*a* is a typical digital logic circuit. The effects of noise coupled into the circuit can be seen in Fig. 5.1*b* and *c*. In Fig. 5.1*b* a small noise pulse is present on the clock line, but the flip-flop does not respond as it does to the greater pulse in Fig. 5.1*c*.

If this circuit were part of some decision-making system, the condition in Fig. 5.1*c* would produce erroneous answers—usually worse than no answer at all. The noise present in Fig. 5.1*b*, while undesirable, is not really harmful.

The origins and characteristics of such noise are considered, along with pertinent device characteristics. Methods for eliminating or counteracting the effects of noise in logic systems are presented.

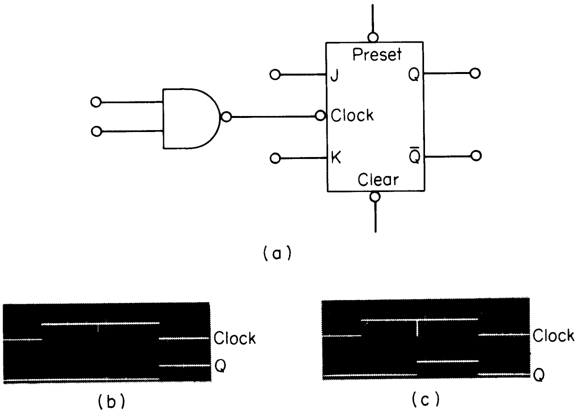


Fig. 5.1. Typical logic circuit with noisy inputs.

5.1 NOISE TYPES AND CONTROL METHODS

Several types of noise must be dealt with in logic systems. The following classification of noise is used:

1. External noise—environmental noise radiated into the system. Such noise results from circuit breakers, brush motors, arcing relay contacts, or magnetic-field-generating activity.
2. Power-line noise—noise coupled through the a-c or d-c power distribution system. Initial sources are frequently the same as those of external noise.
3. Cross talk—noise induced into signal lines from adjacent signal lines.
4. Signal-current noise—noise generated in stray impedances throughout the circuit from the flow of required signal currents.
5. Transmission-line reflections—noise from unterminated transmission lines that cause ringing and overshoot.
6. I_{CC} current spikes—noise caused by switching of the totem-pole output stage.

In general, these noise categories may be dealt with according to the following:

NOISE TYPES	CONSIDERATIONS
External	Shielding
Power line	Grounding
Cross talk	Decoupling
Signal current	
Transmission line	Device properties
I_{CC} spikes	

5.2 SHIELDING

Electrical equipment must function in an extremely noisy environment in addition to its own internally generated noise. Noise pulses may come from many sources, but will be either an electrostatic field or electromagnetic field or both. The noise wavefront must be prevented from entering the equipment. These noise fields are usually changing at a rapid rate; thus, the shield required to exclude them may be quite small. For effective exclusion, the sensitive circuitry must be completely shielded. Although aluminum or similar materials may be effective in stopping electrostatic noise, only a ferrous metal can successfully protect equipment against magnetic fields. It is helpful to connect the system to a good earth ground, but the shield system must be complete and must be grounded to the system ground; otherwise the shield may couple the noise into the system.

External noise may be conducted into the system by the power lines. Decoupling and filtering of these lines should be standard design procedure.

Fast systems of logic, because of their greater bandwidth, are generally assumed to be more susceptible to externally generated noise than are slower systems. This assumption contains some truth, but it is not so relevant as it is widely believed to be. A slow logic system is less susceptible to noise than a faster one only if it is slow *compared with* the noise signals. One of the fastest of the industrial noise generators is the silicon-controlled rectifier (SCR, Thyristor, etc.). These devices produce rise times of about 1 μs. Edges as slow as this are fast when compared with even the slowest types of integrated-circuit logic elements. For example, MOS is probably the slowest IC logic, and yet it has a propagation delay of about 0.5 μs and is as susceptible to industrial noise as the fastest TTL is. Actually, TTL has such low impedance that it is less susceptible to noise than MOS or any other logic system.

5.3 GROUNDING AND DECOUPLING

In the generation of internal noise the total propagation delay is of secondary importance; it is the actual transition time that determines the amplitude/frequency spectrum of the generated signal at the higher harmonics. Application of the Fourier integral to Series SN54/74 waveforms shows frequency components of significant amplitude exceeding 100 MHz. In consideration of the frequency spectrum generated when an SN54/74 device switches, it is obvious that a system using these devices must receive consideration from an RF viewpoint if trouble is to be avoided, even though the repetition rates may be only a few hertz. The transient currents generated by charging capacitance, changes in levels of direct currents, line driving, etc., must be considered. For example, a gate driving a transmission line may be represented by a voltage source E having an output impedance Z_s connected to an impedance Z_0, and loaded with resistance R (Fig. 5.2).

From transmission-line theory,† line termination is not a factor in drive current until after a reflected pulse returns from the termination to the transmitting device. In a practical TTL circuit the line termination must be high relative to the Z_0 of the line. Assume the voltage E is 5 V in amplitude, the Z_s of the source is 50 Ω, and the Z_0 of the line is also 50 Ω. When E makes the transition from 0 V to 5 V, the voltage across the input of the line will be, by Ohm's law,

$$V_{in} = E\frac{Z_o}{Z_s + Z_o} = 2.5 \text{ V}$$

Now for the 50-Ω line to become charged, a current must flow onto the line equal to

---

†John L. Stewart, "Circuit Analysis of Transmission Lines," pp. 23–42, John Wiley & Sons, Inc., New York, 1958.

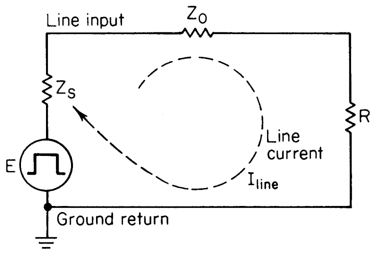


Fig. 5.2. Simplified circuit of a gate driving a loaded transmission line.

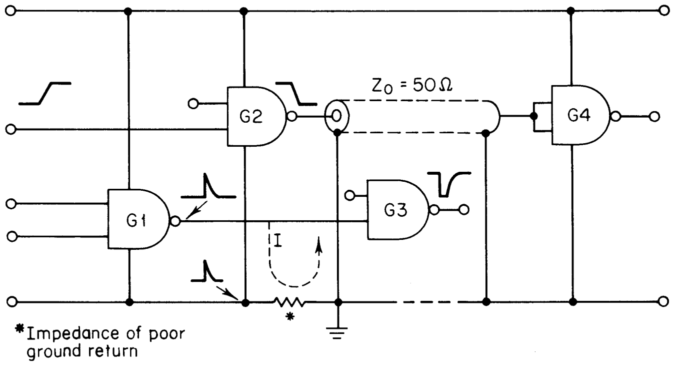


Fig. 5.3. Noise generation due to poor transmission-line returns.

$$\frac{I_{line} = V_{in}}{Z_o} = \frac{2.5}{50} = 50\ \text{mA}$$

This current also flows in the ground return, in this case the transmission-line ground. If the line and return are originated and terminated close to the driving and receiving devices, there is no discontinuity in the line. Where the ground is poorly returned, the currents flowing see the discontinuity in the cable as a high impedance, and a noise spike is generated. Viewed in a slightly different manner, the ground current sees a low impedance and a current cancellation if the ground is properly carried through; if not, it sees a high impedance. See Fig. 5.3 for a specific example. Assume the gate driving the line is switched from the 1 to the 0 state. Current flows as indicated by the arrow marked *I*. Since the line is improperly returned to the driver, a pulse is developed across the impedance. A possible consequence is the false output of gate 3.

If the ground return is properly made, desirable results are obtained in that the impedance discontinuity is eliminated and current cancellation occurs at the ground point. Undesirable voltage spikes are then eliminated. Two rules to reduce transmission-line current effects to acceptable levels in TTL systems may be set forth. These are empirical rules found to be effective (see Figs. 5.4 and 5.5).

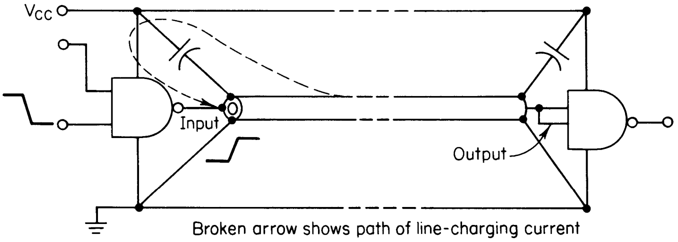


Fig. 5.4. Ideal transmission-line current handling.

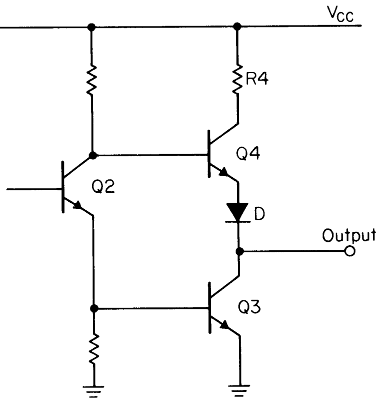


Fig. 5.5. TTL totem-pole output stage for standard and low-power 54/74.

1. Carry twisted pair and coaxial returns to a good ground termination close to the driving and receiving devices. Figure 5.4 illustrates the necessity for this practice.
2. Decouple V_{CC} of line-driving and line-receiving gates close to the device with a 0.1-μF disk ceramic capacitor (see Fig. 5.4).

As the logic devices change state, current levels change owing to different device currents required in each state, to external loading, to transients caused by charging and discharging capacitive loads, and to conduction overlap in the totem-pole output stage. When a gate changes states, its internal current changes from I_{CCH} to I_{CCL} (these values are stated on the data sheet for each device). Also any capacitance, stray or otherwise, must be charged or discharged for a logic state change. The capacitance must be charged by a current:

$$I_c = C\frac{dv}{dt}$$

If we know the total stray capacitance on a gate output, the logic level voltage excursion, and associated rise or fall times, then the ideal-case instantaneous current during the transition can be calculated.

From this equation, it can be seen that the current transient for charging load capacitance will increase with the higher-speed TTL circuits. Therefore, the 54/74L will have the lowest transient current, and the 54/74S will have the highest. Another parameter that should be considered is the value of $R4$ (Fig. 5.5). $R4$ acts as a limit on the charging current; in this regard also, the low-power TTL series has the lowest current transient.

The current required for charging C_L shown in Fig. 5.6 is supplied by the V_{CC} supply when the transition is from logical 0 to logical 1 at the output of $G1$. When the output of $G1$ goes from logical 1 to 0, the capacitance C_L is shorted to ground by transistor $Q3$ (Fig. 5.5) and so has no effect on I_{CC}.

A characteristic common to all TTL totem-pole output stages contributes an additional current transient when the output changes from a logical 0 to a 1. This

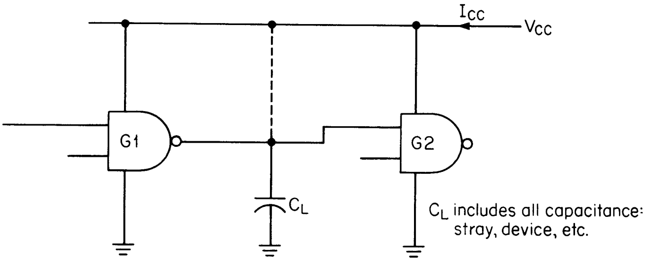


Fig. 5.6. Circuit showing effective capacitive loading.

transient or spike is caused by the overlap in conduction of the output transistors $Q3$ and $Q4$ of Fig. 5.6. The situation arises because $Q4$ can turn on faster than $Q3$ can turn off. This allows a direct circuit for the supply voltage to ground which is limited by R_{CS} of $Q3$ and $Q4$ and $R4$. The peak current can be estimated by

$$I_{CC(\max)} = \frac{V_{CC} - V_D - V_{CE\text{sat}(Q3)} - V_{CE\text{sat}(Q4)}}{R4}$$

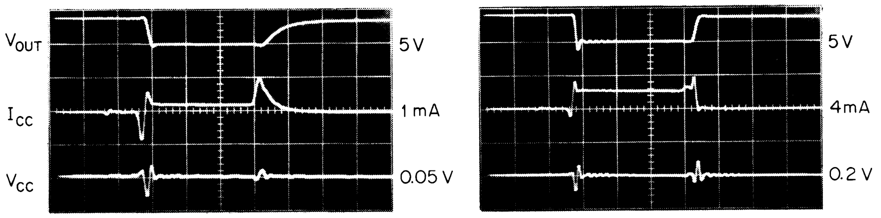
The total charge that flows owing to conduction overlap is determined by the equation above and the duration of the overlap. As would be expected, the low-power TTL series has the lowest current spikes and the longest duration because of the slower speeds. Conversely, the high-speed TTL series exhibits higher current spikes of shorter duration.

The total I_{CC} switching transient is then a combination of these three major effects: difference in I_{CCH} and I_{CCL}, the charging of load capacitance, and conduction overlap. Tests were made to show these effects, and the results are given in Figs. 5.7 and 5.8. First (Fig. 5.7), each of the four types of Series 54/74 was tested with no load (the scope probe was connected to the output *only* when measuring V_{out}, and the photos were double-exposed). This was intended to approximate the effects of conduction overlap isolated from the transient caused by charging load capacitance. Notice that different scales are used in certain of the comparisons. The results are almost as predicted: the SN74L00 has the lowest transient, and the SN74H00 has the highest. But the SN74S00 should be higher still, since it is the fastest circuit. The reason for the decrease can be explained by referring to Fig. 2.4. The additional transistor $Q3$ and associated resistors give more closely matched switching times for the output transistors $Q4$ and $Q6$, resulting in a smaller transient even though the typical rise time is 3 ns compared to 9 ns for the SN54/74.

The second series of tests (Fig. 5.7) covers a capacitive load of 50 pF (refer to Fig. 5.9 for connection). For this test, all the I_{CC} transient peaks increase in amplitude and width. It is interesting to note, in comparing the SN74S and SN74H, that the transients for the two are more nearly equal, because the part of the transient due to the added C_L now predominates over the contribution made by the conduction overlap effect. Voltage spikes on V_{CC}, measured at the IC package, are also increased, owing to the larger transient currents. Finally, Fig. 5.8 shows the additional effect of $C_L = 50$ pF plus a typical load used to measure switching times. Also shown are the effects of connecting bypass or decoupling capacitors for V_{CC} at the IC package.

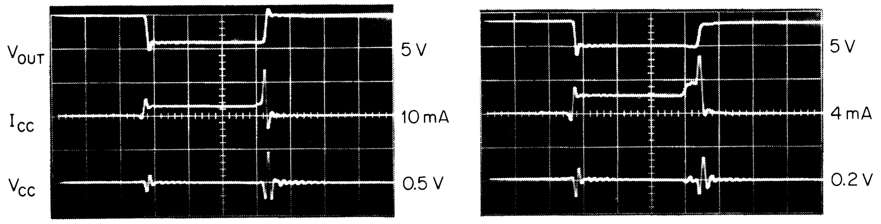
From these tests, one can conclude that the condition to be avoided (in fact, the only one that can be avoided) is unnecessary stray capacitance in circuit wiring. The charging of load capacitance in most cases overshadows the other two effects with respect to noise produced on the V_{CC} line by switching current transients.

The flow paths of these currents have been investigated to determine the grounding and decoupling necessary to counteract their effects. V_{CC} decoupling thus may be approached in two ways; common to both is the requirement to maintain low impedance from the individual circuit V_{CC} to ground. First, the V_{CC} line may be considered a transmission line back to a low-impedance supply. The positive bus may be laminated with a ground bus to form a strip transmission line of extremely



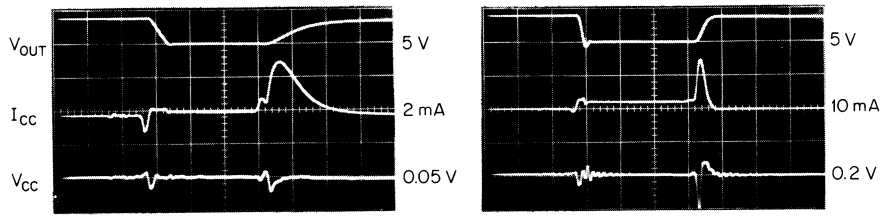
(a) SN74L00 no load

(b) SN7400 no load



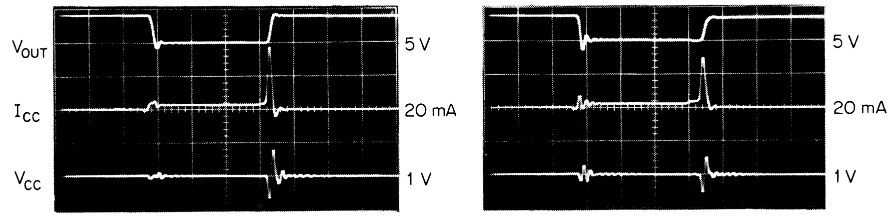
(c) SN74H00 no load

(d) SN74S00 no load



(e) SN74L00 load: CL = 50 pF

(f) SN7400 load: CL = 50 pF

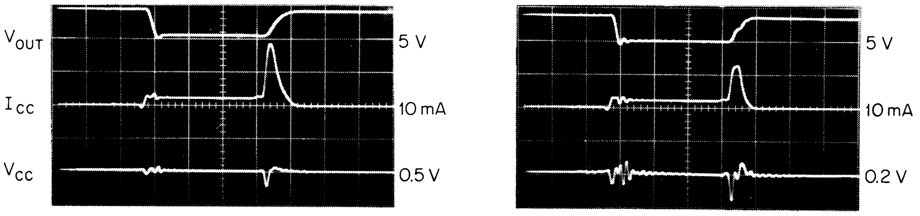


(g) SN74H00 load: CL = 50 pF

(h) SN74S00 load: CL = 50 pF

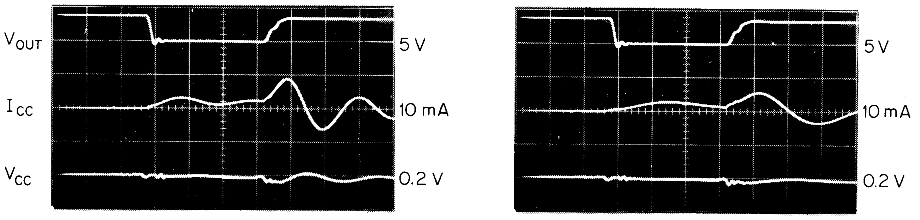
1. $V_{CC} = 5$ V
2. Sweep is 50 ns/division
3. Rise and fall times of input pulse are 10 ns
4. Vertical scales are in units shown per division

Fig. 5.7. I_{CC} transient, comparison of 54/74 types.



(a) Load: $C_L = 50$ pF No bypass

(b) Load: $C_L = 50$ pF Diodes and resistor No bypass



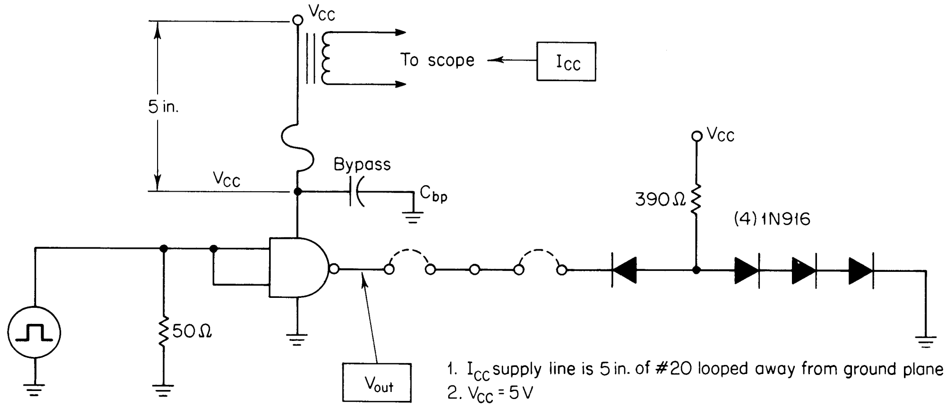
(c) Load: $C_L = 50$ pF Diodes and resistor Bypass of 3,000 pF

(d) Load: $C_L = 50$ pF Diodes and resistor Bypass of 0.01 μF

1. $V_{CC} = 5$ V
2. Sweep is 50 ns/division
3. Rise and fall times of input pulse are 10 ns
4. Vertical scales are in units shown per division

Fig. 5.8. I_{CC} transient for SN7400; comparison for varying loads and bypassing of V_{CC} line.

low impedance as is done in some emitter-coupled logic systems (Fig. 5.10*a*). Such a line may be electrically approximated with lumped capacitances (Fig. 5.10*b*); the inductances are usually a distributed component which must be minimized to lower the Z_0 of the line. An alternative approach is to consider the V_{CC} bus as a d-c connecting element only and provide a low-impedance path near the devices for the transient currents to ground (Fig. 5.11). For effective filtering and decoupling,



1. I_{CC} supply line is 5 in. of #20 looped away from ground plane
2. $V_{CC} = 5$ V

Fig. 5.9. Test circuit for Figs. 5.7 and 5.8.

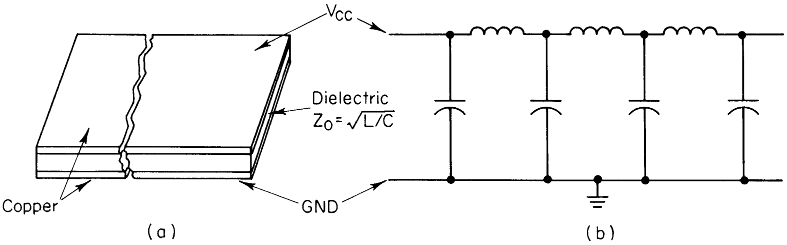


Fig. 5.10. Transmission-line power buses.

the capacitors must be able to supply the change in current for a period of time greater than the pulse width of this current. Since the problem is one of essentially d-c changes due to logic state changes coupled with transients associated with the changes, two different values of time constant must be considered. Capacitors combining the high capacitance required for long periods with the low series reactance required for fast transients are prohibitive in cost and size. A good compromise is the arrangement shown in Fig. 5.12. Typical component values may be found as follows. For the RF capacitor $C2$, assume the parameters to have common values such as

$$\Delta I_{CC} = 50 \text{ mA}$$

$$\Delta V \leq 0.1 \text{ V}$$

$$\Delta T = 20 \text{ ns}$$

then

$$C2 = \frac{\Delta I_{CC}}{\Delta V/\Delta T} = \frac{50 \times 10^{-3}}{0.1/(20 \times 10^{-9})} = \frac{(50)(20) \times 10^{-12}}{0.1}$$

$$= 10{,}000 \times 10^{-12}$$

$$= 0.01 \ \mu\text{F}$$

The same approach may be used for the low-frequency capacitor $C1$. However, the factor ΔT, which was merely a worst-case transient time when calculating $C1$, now becomes a bit hazy. An analysis of the current cycling on a statistical basis is the best approach in all but the simplest systems. The recommended procedure is to use brute force and decouple with 10 to 50 μF.

A discrete inductance of 2 to 10 μH is sometimes added for further decoupling. However, its benefits are questionable, and its usefulness should be evaluated for

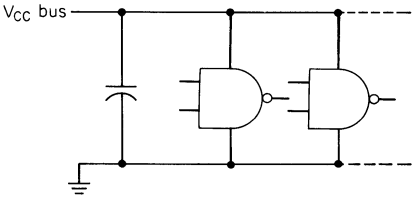


Fig. 5.11. Capacitive storage V_{CC} system.

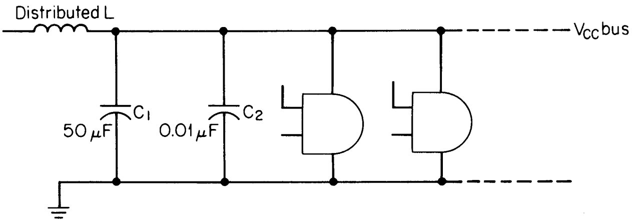


Fig. 5.12. Commonly used power distribution and decoupling system.

the individual system. The low-pass filter formed must be capable of keeping the transients confined and off the distribution bus. The possibility of resonance in the inductor or LC combination must be considered.

Noise spikes on the V_{CC} line that do not force the gate output below the threshold level are not serious. Downward spikes as great as 3 V have been tolerated on the V_{CC} line without propagating through the logic system. Thus the system designer can be greatly relieved on the subject of V_{CC} noise if even minimal consideration is given to the subject.

Ground noise is another matter. Pulses on the ground line can quite readily exceed the noise threshold. Only if a good ground system is maintained can this problem be overcome. If proper attention is paid to the ground system, problems seldom arise.

The concept of a common ground plane structure as used in RF and high-speed digital systems is quite different from that of the common ground point as used in low-frequency circuits. The more closely the chassis and ground can approach being an integral unit, the better the noise suppression characteristics of the system. Consequently all parts of the chassis and ground bus system should be bound tightly together electrically and mechanically. Floating or poorly grounded sections not only break the integrity of the ground system, but may actually act as a noise distribution system.

The matter of grounds and decoupling on printed boards requires some attention here. The most desirable arrangement would be a double-clad board with one side carrying the interconnections and the other a solid ground plane. Where component density prohibits this, the ideal should be relaxed only as far as necessary. Cross talk can become a problem on large boards; an adequate board ground mesh will go far in reducing cross talk as well as ground noise. Some suggestions for board grounds where a plane is not practical are:

1. Make the ground strap as wide as possible everywhere, even though this may mean its width varies radically.
2. Form a complete loop around the board; bring both sides of the board through separate pins to the system ground.

The V_{CC} line can provide part of the ground mesh on the board, provided it is properly decoupled. For a TTL system a good rule of thumb is 0.01 μF per synchronously driven gate and at least 0.1 μF per 20 gates regardless of synchronization. This capacitance may be lumped, but it is more effective if distributed over the board. A good rule to follow is to permit no more than 5 in. of wire between any two package V_{CC} points. RF-type capacitors for decoupling are mandatory; disk ceramics are a good choice. It is sometimes a good practice to decouple the board from the external V_{CC} line with a 2.2-μH inductor and a 10- to 50-μF capacitor as stated earlier, but this is optional, and the RF capacitors will still be required. In addition, it is strongly recommended that gates driving long lines have the V_{CC} supply decoupled at the gate V_{CC} terminal and that the capacitor ground, device ground, and transmission-line ground be brought to a common point. The reasons for this should be more obvious after studying the currents involved in switching and line driving.

5.4 CROSS TALK

When currents and voltages are impressed on a connecting line in a system, it is impossible for adjacent lines to remain unaffected. Static and magnetic fields interact and opposing ground currents flow, creating linking magnetic fields. These cross-coupling effects are lumped together as *cross talk*.

Signal lines are grouped into three categories: coaxial lines, twisted-pair lines, and straight wire lines. Because of the low impedance and shielding characteristics of coaxial cable, coaxial cable cross talk is minimal and is no problem with TTL.

Figure 5.13 is adequately descriptive of practical types of signal transmission lines. The mutual reactances L_m and C_m which form the noise coupling paths, and the line parameters L_s and C_g which govern the Z_0 of the line, will vary with the type of line used. Since cross talk is a function of the ratio of the mutual impedances to the line characteristic impedances, the selection of transmission-line type must be at least partially a factor in cross talk considerations.

Direct-wired connections are the simplest and cheapest approach, but they are also the poorest for noise rejection. It is possible to use direct leads up to 10 in., and up to 20 in. if the lead is routed close to ground and not cabled tightly together with similar leads.

When the length of the signal line increases, the line impedance is seen by the driving and receiving gates. As shown in Fig. 5.14, a pulse sent along the *sending line* *G*3-*G*4 will be coupled via coupling impedance Z_c onto the *receiving line* *G*1-*G*2,

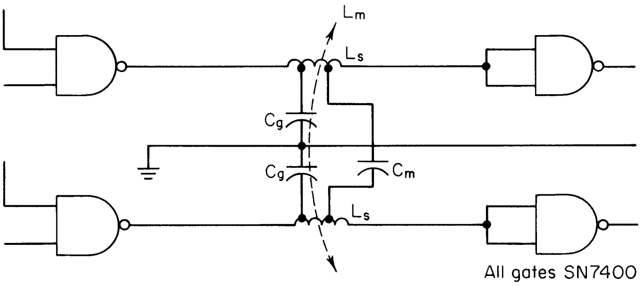


Fig. 5.13. Coupling impedances involved in cross talk.

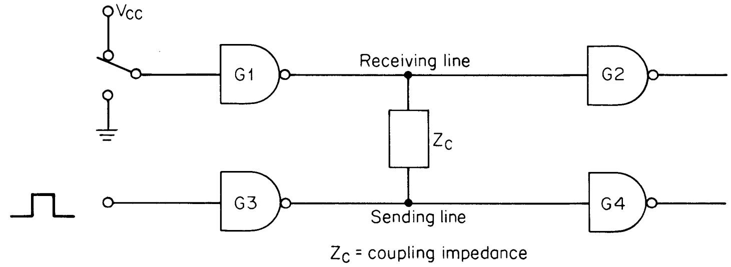


Fig. 5.14. Illustration of capacitive cross talk between two signal lines.

which can be in either of the two logic states. The extent to which cross talk will occur depends on the type of lines used and their relationship to each other.

The voltage impressed on line $G3$-$G4$ by gate $G3$ is

$$V_{SL} = \frac{V_{G3} Z_o}{R_{S3} + Z_o}$$

where V_{G3} = open-circuit logic voltage swing generated by gate $G3$
R_{S3} = output impedance of $G3$
Z_o = line impedance
V_{SL} = voltage impressed on the sending line

The relationship for the above equation is illustrated in Figs. 5.15 and 5.16.

The coupling from the sending to the receiving line can be represented by taking the coupling impedance Z_c into account. An equivalent circuit to represent the coupling from the sending line to the receiving line is shown in Fig. 5.16.

Two Z_o loads are shown at the receiving line to represent the fact that, from the theoretical lumped coupling point in the receiving line, there are two lines in parallel, one going to gate $G1$ and one to gate $G2$. The voltage V_{RL} developed across the receiving line is therefore

$$V_{RL} = V_{SL} \frac{Z_o}{2(Z_c + Z_o)}$$

Since the input impedance of gate $G2$ will almost invariably be large compared with Z_o, the full open-circuit output voltage of the receiving line will appear at the input of $G2$. Therefore,

$$V_{in(2)} = 2 V_{RL}$$

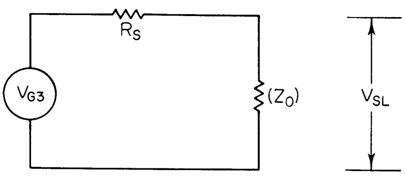


Figure 5.15

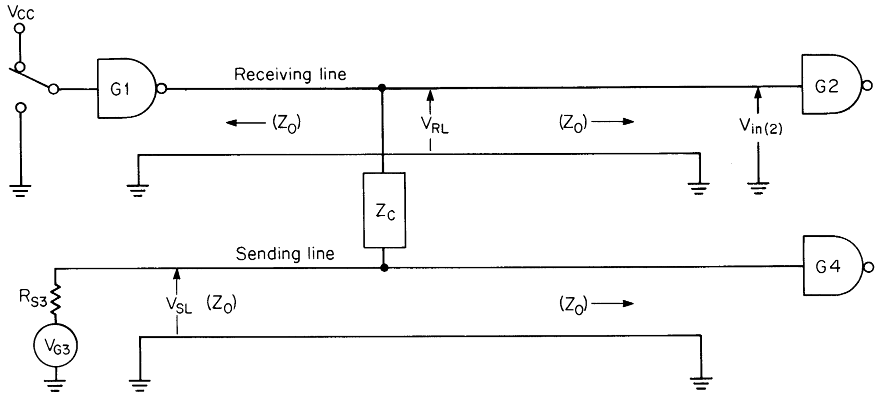


Figure 5.16

Thus

$$\frac{V_{in(2)}}{V_{G3}} = \frac{Z_o \cdot Z_o}{(Z_c + Z_o)(R_{S3} + Z_o)}$$

If $R_{S3} \ll Z_o$, this expression may be simplified to

$$\frac{V_{in(2)}}{V_{G3}} = \frac{1}{1 + Z_c/Z_o}$$

which will be the cross talk noise-to-signal ratio of this transmission system.

The worst case for signal-line cross talk occurs when sending and receiving lines are close together but widely separated from a ground return path. The lines then have a high characteristic impedance and a low coupling impedance Z_c. For example, if the lines consist of $\frac{1}{25}$-in. conductors spaced $\frac{1}{30}$ in. apart and farther than 0.78 in. from any ground conductors, the impedances will be $Z_o \approx 200\ \Omega$ and $Z_M \approx 80\ \Omega$. The signal-to-noise ratio will then be

$$\frac{V_{in(2)}}{V_{G3}} = \frac{1}{1 + 80/200} = 0.71$$

This is clearly unsatisfactory, for no normal logic circuit has a noise margin greater than one-third the logic swing. Such potential cross talk can be avoided by preventing such close spacing of conductors.

If in this example a ground plane were introduced, spaced $\frac{1}{25}$ in. from each conductor, the impedances become $Z_o \approx 50\ \Omega$ and $Z_c \approx 125\ \Omega$. Substituting these values yields

$$\frac{V_{in(2)}}{V_{G3}} = \frac{1}{1 + 125/50} = 0.29$$

Although this degree of cross talk would probably not be disastrous, it provides little safety margin.

If the sending and receiving lines were twisted pairs laid closely side by side, the impedances would be $Z_0 \approx 80\ \Omega$ and $Z_c \approx 400\ \Omega$. The cross talk noise-to-signal ratio would then be

$$\frac{V_{in(2)}}{V_{G3}} = \frac{1}{1 + 400/80} = 0.17$$

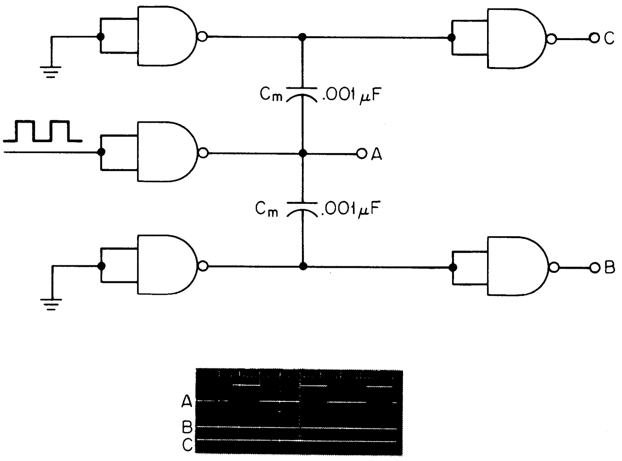
This is safe and practical for all conventional logic circuits including 54/74 TTL.

Mutual coupling can be reduced by using coaxial cable or shielded twisted pairs. When mutual inductance and capacitance are decreased, however, line capacitance is increased, imposing restrictions on the driver. Coaxial cable combines very high mutual impedance with low characteristic impedance, and shielding as well. It effectively eliminates cross talk, but it is necessary in only the noisiest environments. Twisted pairs are usually adequate, and they are typically less expensive and easier to work with.

Fortunately, much of the noise that may be coupled in is subdued by the low output impedances of 54/74 TTL. An indication of the superior noise rejection characteristics of SN54/74 TTL can be observed in the circuit in Fig. 5.17. Although C_m is large (0.001 μF) and C_g so small that it can be ignored, capacitive cross talk still does not appear. This leaves the designer only inductive coupling effects to deal with.

5.5 TRANSMISSION-LINE REFLECTIONS

When the interconnections used to transfer digital information become long enough so that line propagation delay is equal to or greater than the pulse transition times, the effects of reflections must be considered. These reflections are created because most TTL interconnections are not terminated in their characteristic im-



All gates SN7400N Vert: 5 V/cm
Horiz: 0.5μs/cm

Figure 5.17

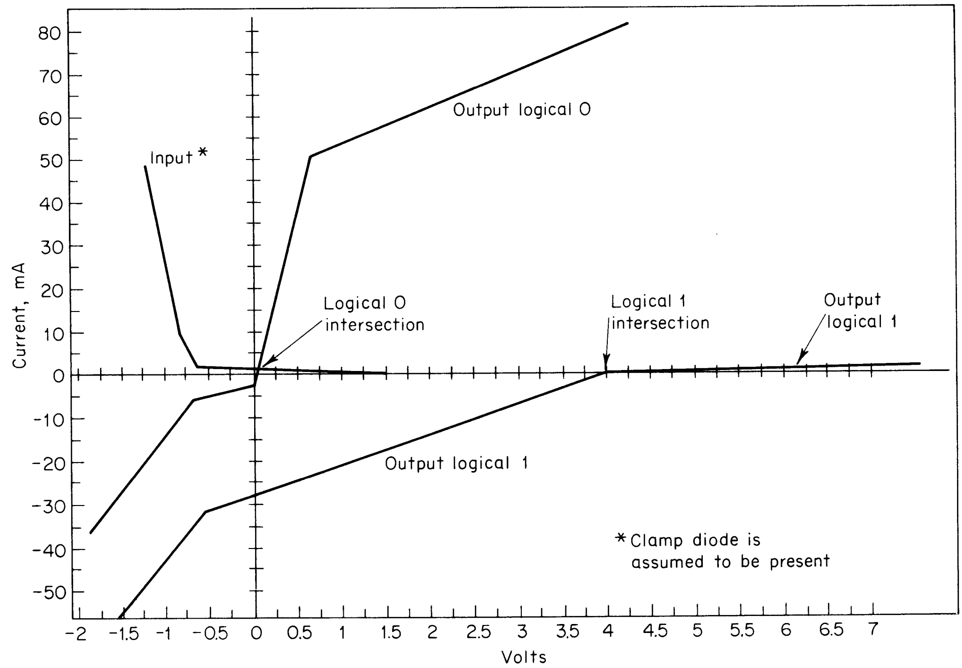
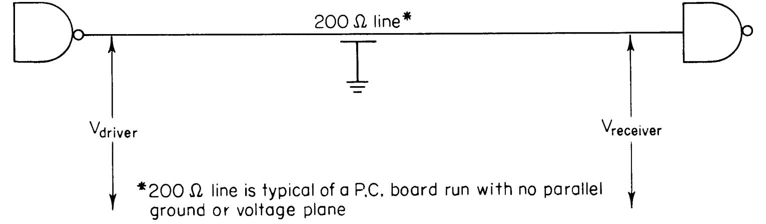


Fig. 5.18. Output and input characteristics of a typical Series SN54/74 device at $T_A = 25°\text{C}$ and $V_{CC} = 5.0$ V.

pedance. Reflections lead to such things as reduced noise margins, excessive delays, ringing, and overshoot. Some method must be used to analyze these reflections. Basic transmission-line equations are applicable but unwieldy, since neither the gate's input nor output impedance is linear. Transmission-line characteristics of TTL interconnections can best be analyzed by a simple graphic technique.

Figure 5.18 shows piecewise linear plots of a gate input and both (logical 0 and 1) conditions of the output for a typical SN54/74 device. The output curves are plotted in the normal way with positive slopes, but the input is inverted, since it will be at the receiving end of a transmission line. The logical 0 and 1 intersections are indicated on the plot. These points are the steady-state values which will be observed on a lossless transmission such as shown in Fig. 5.19.



*200 Ω line is typical of a P.C. board run with no parallel ground or voltage plane

Fig. 5.19. Typical TTL interconnection.

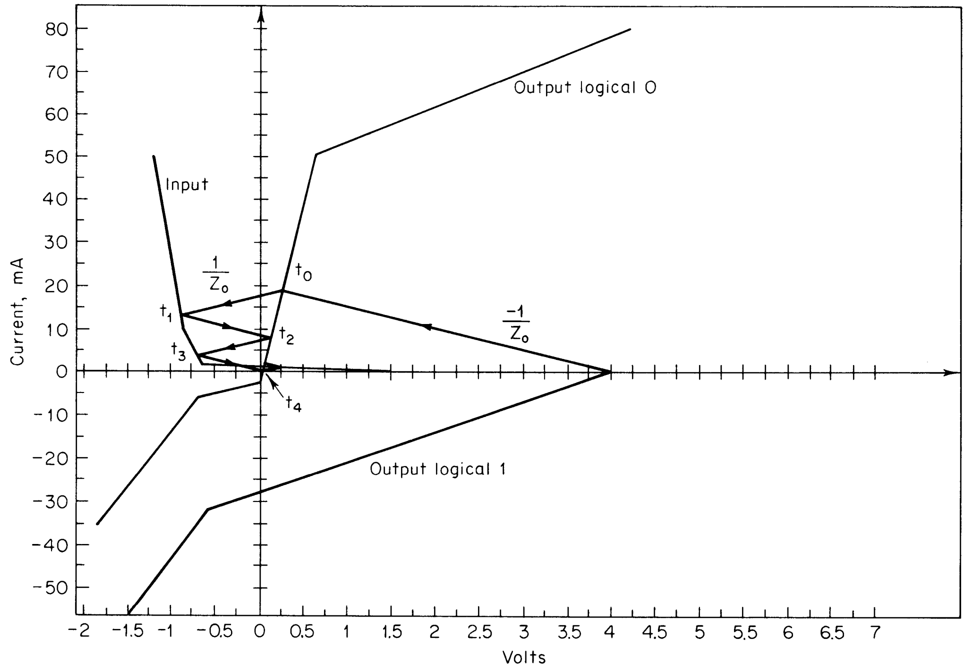


Fig. 5.20. Logical 1-to-0 transition, $Z_o = 200\ \Omega$.

Figure 5.19 shows a typical TTL interconnection. Open wires will usually have characteristic impedances of 150 to 200 Ω, depending upon the amount of capacitive coupling to ground. To evaluate a logical 1-to-0 transition, Fig. 5.20 is used. The line $-1/Z_o$ ($Z_o = 200\ \Omega$) which represents the transmission line is superimposed on the output characteristic curves. Since evaluation of a logical 1-to-0 transition is desired, the $-1/Z_o$ line starts at the point of intersection of the impedance curves of the input and output for a logical 1 condition; this point is labeled t_0. The slope

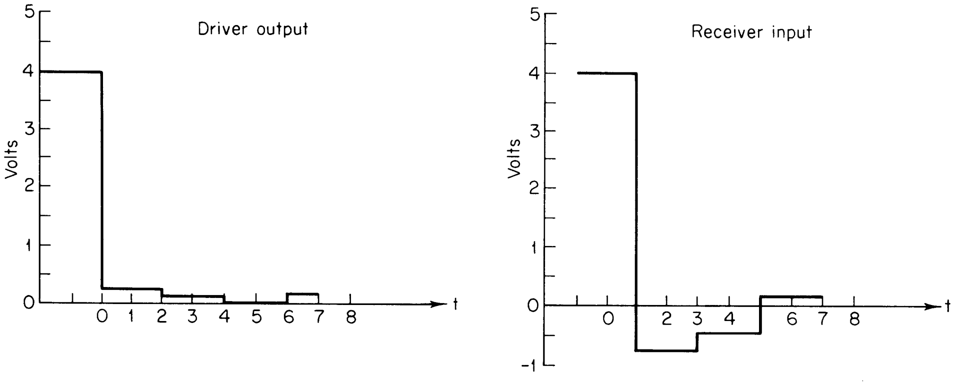


Fig. 5.21. Voltage versus time for $Z_o = 200\ \Omega$ and a logical 1-to-0 transition.

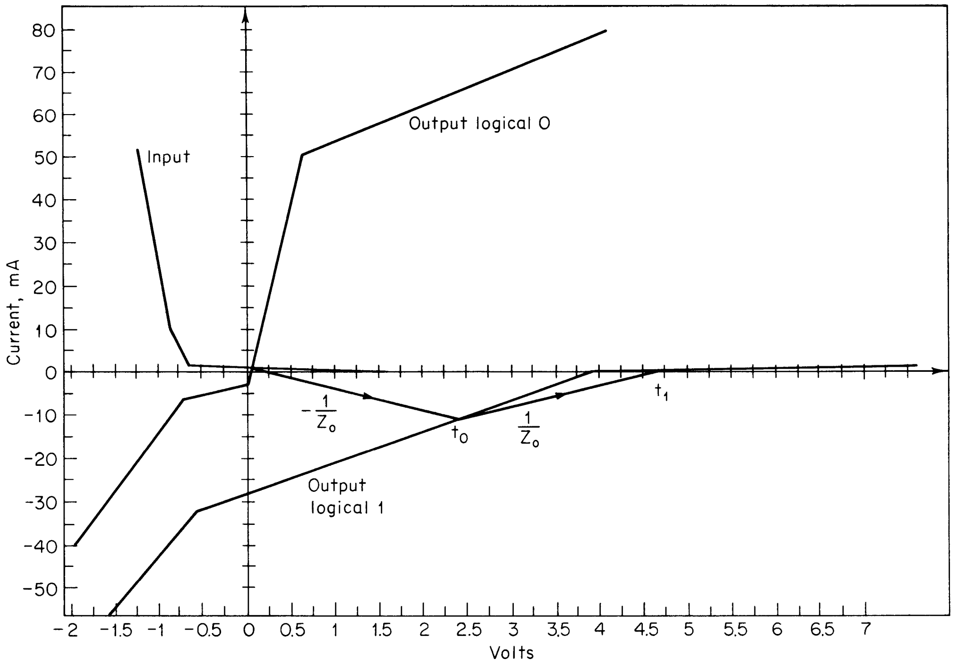


Fig. 5.22. Logical 0-to-1 transition, $Z_o = 200\ \Omega$.

$-1/Z_o$ then proceeds toward the logical 0 output curve. At time t_0, the driver output voltage is determined by the intersection of $-1/Z_o$ and the output 0 curve ($\approx +0.25$ V). The transmission-line slope now becomes $+1/Z_o$ and is drawn toward the input curve. At time t_1 ($t_{n+1} - t_n$ = time delay of line), the receiving gate sees -0.75 V. Now the line slope changes back to $-1/Z_o$, and the graph continues as the intersection of the input and output curves for a logical 0 is approached. Figure 5.21 plots driver and receiver voltages versus time in this case.

A logical 0-to-1 transition is treated in approximately the same manner (Fig. 5.22). The line $-1/Z_o$ now starts at the intersection for a logical 0. At time t_0, the driver's output rises to $+2.4$ V, and at time t_1, the receiving gate's input goes to $+5.0$ V. Ringing then occurs on the line because both ends are now terminated in a high impedance. This ringing decreases in amplitude as a steady-state condition is approached. Both the output and input voltages are plotted in Fig. 5.23 for this case.

In Figs. 5.24 to 5.31, graphic analysis is applied to 50- and 100-Ω transmission lines. Notice that a decrease in line impedance does not significantly change the logical 1-to-0 transition characteristics. This is owing to the low driver output impedance in the 0 state and the presence of an integral clamping diode on the input of the receiving gate. Line impedance does, however, significantly alter the 0-to-1 transition characteristics. As the line impedance increases, overshoot at the receiving end also increases. This overshoot should be limited to no more than $+5.5$ V to avoid emitter-to-emitter breakdown of the receiving gate. Low impedance

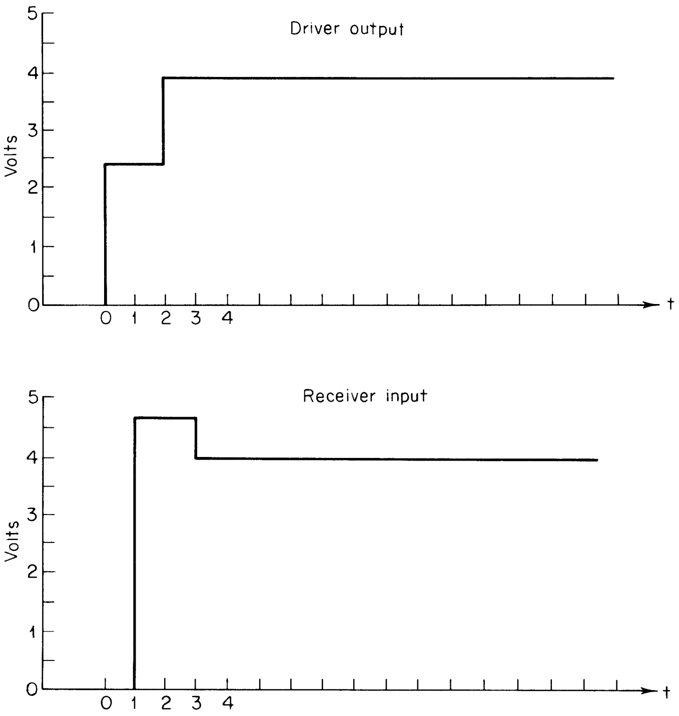


Fig. 5.23. Voltage versus time for $Z_o = 200\ \Omega$ and a logical 0-to-1 transition.

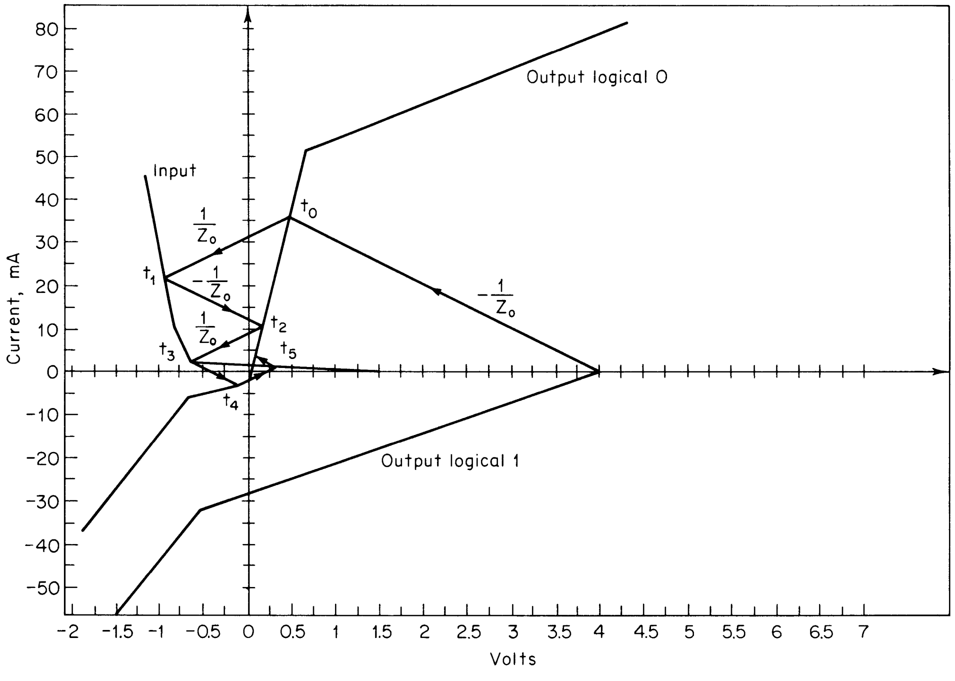


Fig. 5.24. Logical 1-to-0 transition, $Z_o = 100\ \Omega$.

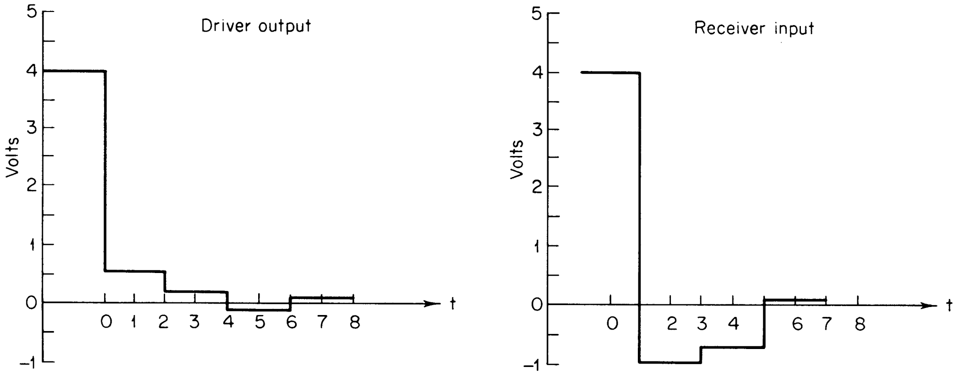


Fig. 5.25. Voltage versus time for $Z_o = 100\ \Omega$ and a logical 0-to-1 transition.

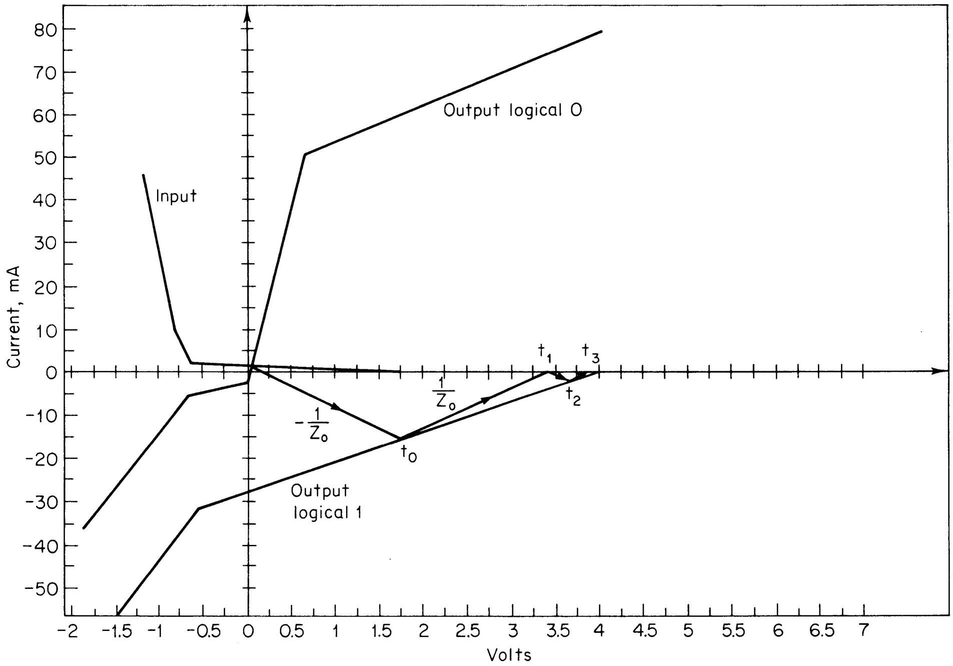


Fig. 5.26. Logical 0-to-1 transition, $Z_o = 100\ \Omega$.

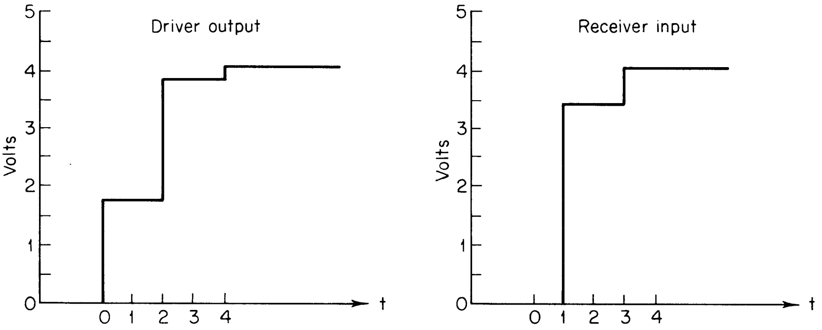


Fig. 5.27. Voltage versus time for logical 0-to-1 transition and $Z_o = 100\ \Omega$.

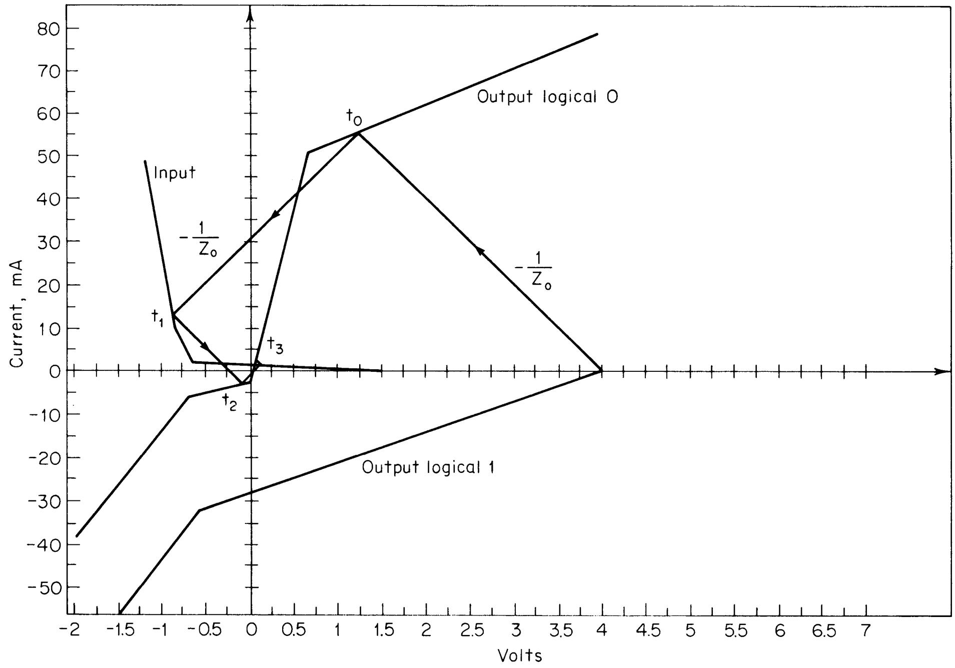


Figure 5.28

lines significantly affect noise margins of gates placed as shown in Fig. 5.32. Note that receiver *B* will see only +1.25 V for two time delays of the transmission line. Its output can therefore not be guaranteed until time t_2, or two transmission-line delays.

The scope photographs of Fig. 5.33 were taken to show the effectiveness of the graphic techniques. In most cases, the calculated and experimental values of voltage steps agree within reason. The ringing that appears for the open wire is not imme-

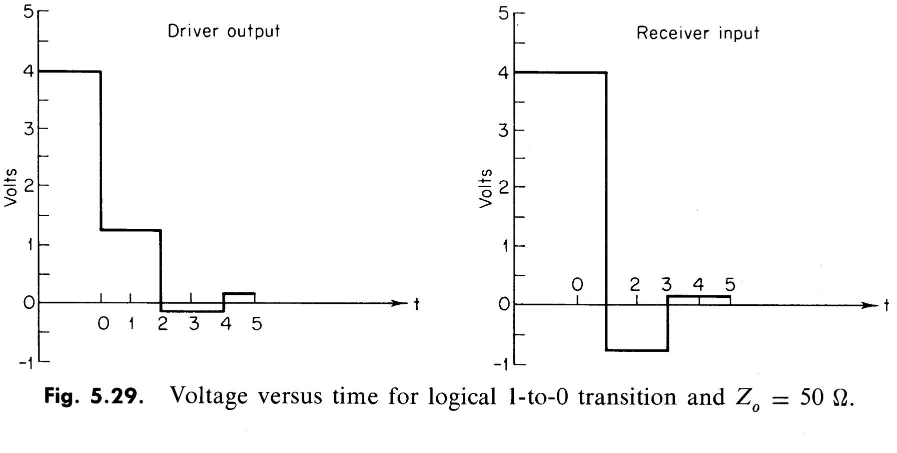


Fig. 5.29. Voltage versus time for logical 1-to-0 transition and $Z_o = 50\ \Omega$.

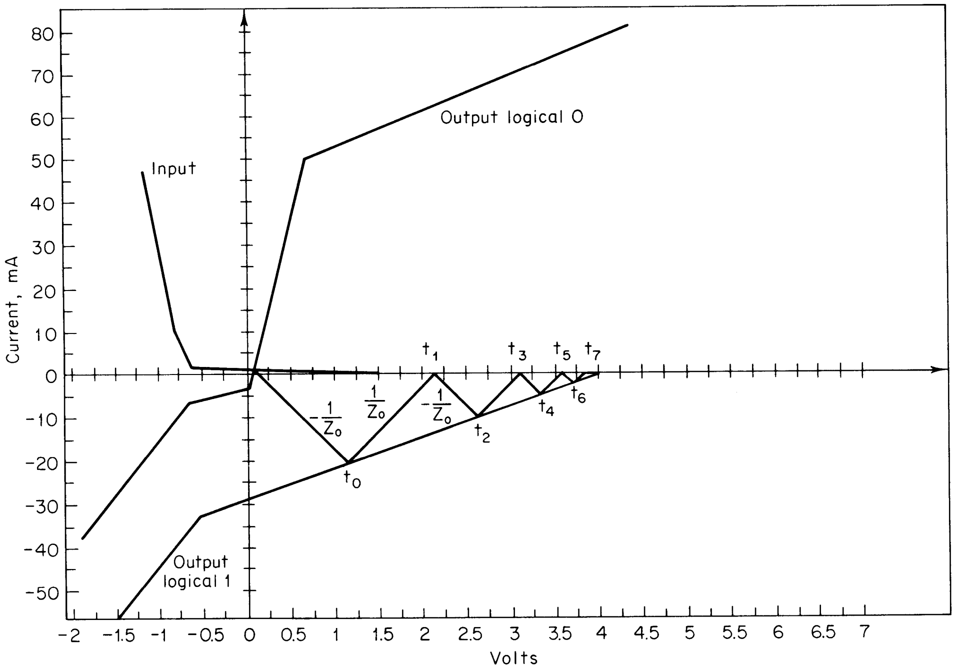


Figure 5.30

diately obvious, but this is because the input and output curves in this region lie practically along the positive horizontal axis. At the scale used for the graphic analysis, it is difficult to go much beyond the first few reflections. As stated before, the graphic analysis is idealized; stray capacitance and inductance are not considered.†

---

†For additional information see John A. De Falco, Reflections and Crosstalk in Logic Circuit Interconnections, *IEEE Spectrum,* July, 1970, and Robert S. Singleton, No Need to Juggle Equations to Find Reflections—Just Draw Three Lines, *Electronics,* Oct. 28, 1968.

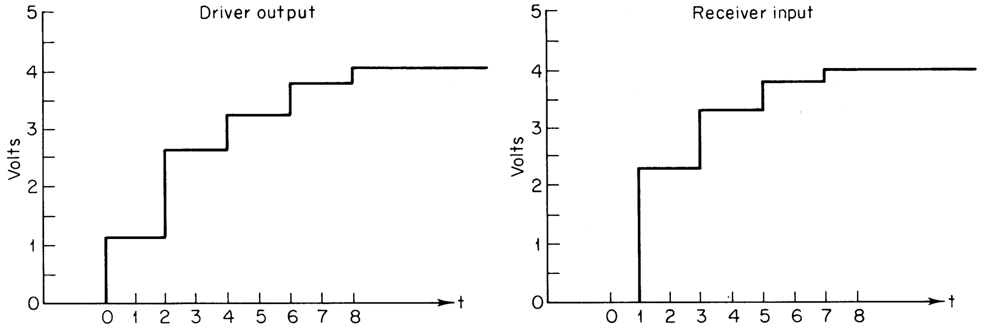


Fig. 5.31. Voltage versus time for logical 0-to-1 transition and $Z_0 = 50\ \Omega$.

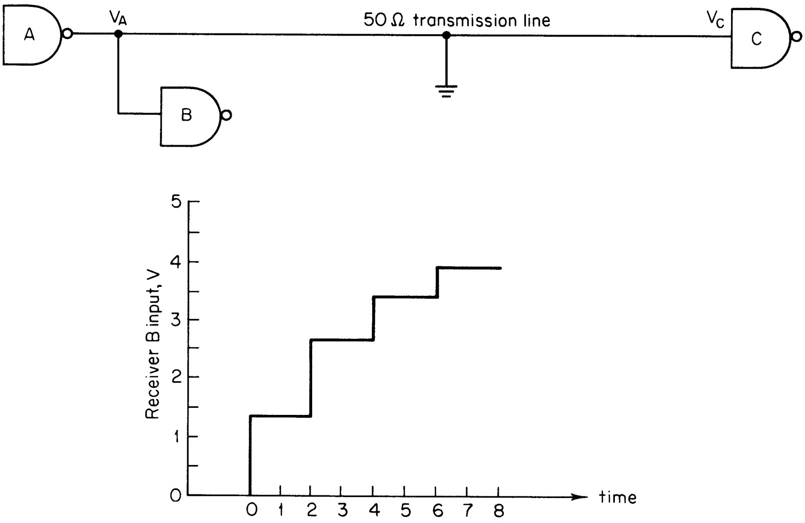
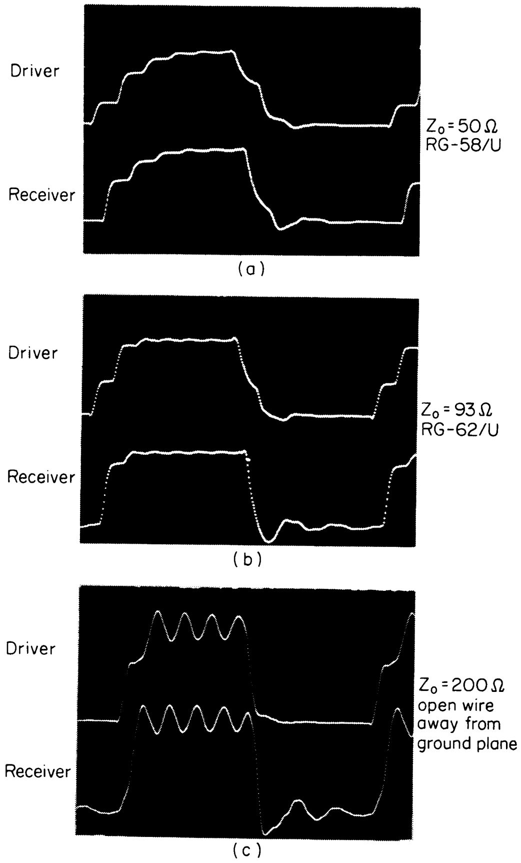

A
B
C
V_A
V_C
50 Ω transmission line
Receiver B input, V
0
1
2
3
4
5
time
0 1 2 3 4 5 6 7 8

Fig. 5.32. Effect on nearby gate input due to low impedance line.



Scale: 2.0 V per division
50 ns per division

$V_{CC} = 5$ V
$T_A = 25°$ C

Figure 5.33

The following rules have been established for the minimization of transmission-line effects in TTL systems:

1. Use direct wire interconnections that have no specific ground return for lengths up to about 10 in. only. A ground plane is always desirable.
2. Direct wire interconnections must be routed close to a ground plane if longer than 10 in. and should never be longer than 20 in.
3. When using coaxial or twisted-pair cables, design around approximately 100 Ω characteristic impedance. Coaxial cable of 93 Ω impedance (such as Microdot 293-3913) is recommended. For twisted pair, No. 26 or No. 28 wire with thin insulation twisted about 30 turns per foot works well. Higher impedances increase cross talk, and lower impedances are difficult to drive.
4. Ensure that transmission-line ground returns are carried through at both transmitting and receiving ends.
5. Connect reverse termination at driver output to prevent negative overshoot.
6. Decouple line-driving and line-receiving gates as close to the package V_{CC} and ground pins as practical, with a 0.1-μF capacitor.
7. Gates used as line drivers should be used for that purpose only. Gate inputs connected directly to a line-driving output could receive erroneous inputs due to line reflections, long delay times introduced, or excessive loading on the driving gate.
8. Gates used as line receivers should have all inputs tied together to the line. Other logic inputs to the receiving gate should be avoided, and a single gate should be used as the termination of a line.
9. Flip-flops are generally unsatisfactory line drivers because of the possibility of collector commutation from reflected signals.

6

Combinational Logic Design

Boolean algebra, the algebra of logic, is the mathematical framework upon which logic design is based, and it is used in the description, synthesis, and analysis of binary logical functions. Boolean algebra is based upon the concept that logical statements that can be designated as true (1) or false (0) can be made. No assignment of intermediate values is allowed. The following material is developed around the basic logic functions and the means of implementing these functions by using Series SN54/74 logic elements. This discussion is not intended to be a rigorous treatment of Boolean algebra, and no proofs of theorems or postulates are given. The interested reader is referred to the Bibliography at the end of the chapter for more detailed information on this subject.

6.1 BASIC FUNCTIONS

Basic logic functions and operations are the AND function, the OR function, and the NOT operation.

1. AND FUNCTION. An AND function is a 1 if, and only if, all logic variables involved are 1s. This is written for two logic variables A and B as

   $$Y = A \cdot B$$

   and can be represented by a truth table (Fig. 6.1). Normally, the dot denoting the AND function is omitted and $A \cdot B$ is written as AB.
2. OR FUNCTION. An OR function is a 1 if at least one of the variables involved is a 1. For two logic variables A and B, this is written as

   $$Y = A + B$$

   and can be represented by a truth table (Fig. 6.2).
3. NOT OPERATION. This is an inverting operation. It converts the state or value of a function or variable to its complement. Performing the NOT operation on a function or variable which is a 1 results in a 0 and vice versa. Several notations are used to indicate the NOT operation, such as adding

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Figure 6.1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Figure 6.2

asterisks, stars, primes, etc. The most common is a bar over the function or variable, e.g., "NOT A" or the "complement of A" will be written as $\bar{A}$.

6.2 POSTULATES AND THEOREMS

Properties of ordinary algebra that are also valid for Boolean algebra are:

Commutative property: *a:* $AB = BA$
b: $A + B = B + A$

Associative property: *a:* $A(BC) = (AB)C$
b: $A + (B + C) = (A + B) + C$

Distributive property: *a:* $A(B + C) = AB + AC$
b: $A + BC = (A + B)(A + C)$

Boolean algebra is governed by a set of rules that are not, however, always equal to those valid for ordinary algebra.

The rules governing Boolean algebra are laid down in the postulates and the resulting theorems.

POSTULATES

1*a:* $A = 1 \text{ if } A \neq 0$
1*b:* $A = 0 \text{ if } A \neq 1$
2*a:* $0 \cdot 0 = 0$
2*b:* $1 + 1 = 1$
3*a:* $1 \cdot 1 = 1$
3*b:* $0 + 0 = 0$
4*a:* $1 \cdot 0 = 0$
4*b:* $0 + 1 = 1$
5*a:* $\bar{0} = 1$
5*b:* $\bar{1} = 0$

THEOREMS

1*a:* $A + 0 = A$
1*b:* $A \cdot 1 = A$
2*a:* $A + 1 = 1$
2*b:* $A \cdot 0 = 0$
3*a:* $A + A = A$
3*b:* $A \cdot A = A$
4*a:* $\overline{(A)} = \bar{A}$
4*b:* $\overline{(\bar{A})} = A$
5*a:* $A + \bar{A} = 1$
5*b:* $A \cdot \bar{A} = 0$

6a: $\overline{A + B + C + \cdots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \cdots$

6b: $\overline{A \cdot B \cdot C \cdot \cdots} = \bar{A} + \bar{B} + \bar{C} + \cdots$ De Morgan's theorem

7a: $A(A + B) = A + AB = A$

7b: $A + AB = A(A + B) = A$

The postulates and theorems indicated with an (*a*) are the duals of the postulates and theorems indicated with a (*b*). Duality is defined as the changing of all AND signs to OR signs, all OR signs to AND signs, all 1s to 0s, and all 0s to 1s. Duality is one of the governing rules of Boolean algebra, and it can be stated that "if an equation is true, then its dual is also true."

The rule of duality is an extremely valuable tool in logic design, especially for complementing logic (Boolean) functions and expressions. Care is necessary when applying the rule of duality; logic functions as they are generally written contain both explicit and implicit parentheses. These groupings must be maintained when applying the rule of duality. For example,

$$Y = \bar{A}(B + C\bar{D}) + \bar{B}D$$

Now with explicit and implicit parentheses, this equation must be written as

$$Y = \{\bar{A}[B + (C\bar{D})]\} + (\bar{B}D)$$

Application of the rule of duality gives

$$\bar{Y} = \{A + [\bar{B} \cdot (\bar{C} + D)]\} \cdot (B + \bar{D})$$

Omitting the superfluous brackets results in

$$\bar{Y} = [A + \bar{B}(\bar{C} + D)](B + \bar{D})$$

De Morgan's theorem (theorem 6) is a good example of the rule of duality, and it will be obvious that it may also be used in the form

$$A \cdot B \cdot C \cdot \cdots = \bar{A} + \bar{B} + \bar{C} + \cdots$$

$$A + B + C + \cdots = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \cdots$$

6.3 LOGIC EXPRESSIONS

As shown, an AND operation is indicated by a dot, although in most cases this dot is omitted. This notation is used in ordinary algebra to denote multiplication. The original terminology is carried over, and the AND function is in some cases referred to as a *product*. The word "product" loses its original meaning when used in Boolean algebra, however, and serves only to indicate an AND function. Similarly, an OR function, which is indicated by a plus sign, is at times referred to as a *sum*. The terms "product" and "sum" in the sense described have been introduced into Boolean algebra for convenience in discussing and describing logic expressions.

It is often desirable to write logic expressions in an easily examinable form. For this purpose logic expressions may be written as:

1. A *sum of products,* i.e., an OR-ing of AND-ed variables such as:

$$AB + AC + \bar{B}C$$

   in which the products are terms of the expression; or
2. A *product of sums,* i.e., an AND-ing of OR-ed variables such as

$$(P + \bar{Q})(Q + R)(P + \bar{R})$$

   in which the sums are the terms of the expression.

The terms in the expressions above can be expanded to obtain a general form in which each term is a unique combination of *all* variables involved. A sum of products can be expanded by applying theorems 1*b*, 3*b*, and 5*a*. For example,

$$\begin{aligned} Y = AB + AC + \bar{B}C &= A \cdot B \cdot 1 + A \cdot C \cdot 1 + \bar{B} \cdot C \cdot 1 \\ &= AB(C + \bar{C}) + AC(B = \bar{B}) + \bar{B}C(A + \bar{A}) \\ &= ABC + AB\bar{C} + ABC + A\bar{B}C + A\bar{B}C + \bar{A}\bar{B}C \\ &= ABC + AB\bar{C} + A\bar{B}C + \bar{A}\bar{B}C \end{aligned}$$

The general form obtained is called the *minterm canonical form,* sometimes also referred to as the *standard sum.* Its individual terms are usually called *minterms.* Other names used instead of minterm are *minterm canonical term, canonical term, product term,* and *P-term.*

A product of sums can be expanded into the general form by applying theorems 1*a* and 5*b* and the distributive property. For example,

$$\begin{aligned} Y = (P + \bar{Q})(Q + R)(P + \bar{R}) &= (P + \bar{Q} + 0)(Q + R + 0)(P + \bar{R} + 0) \\ &= (P + \bar{Q} + R\bar{R})(Q + R + P\bar{P})(P + \bar{R} + Q\bar{Q}) \end{aligned}$$

Now, using the distributive property, each sum term can be expanded:

$$Y + (P + \bar{Q} + R)(P + \bar{Q} + \bar{R})(Q + R + P)(Q + R + \bar{P})(P + \bar{R} + Q)(P + \bar{R} + Q)$$

It is easier to recognize repeated terms if the variables are all in the same order, so rearranging:

$$Y = (P + \bar{Q} + R)(P + \bar{Q} + \bar{R})(P + Q + R)(\bar{P} + Q + R)(P + Q + \bar{R})(P + \bar{Q} + \bar{R})$$

This is called the *maxterm canonical form* and sometimes the *standard product.* Its individual terms are usually called *maxterms;* other names are *maxterm canonical term, sum term,* and *S-term.*

6.4 SIMPLIFICATION AND MINIMIZATION

Logic expressions and equations describing the functions and operations that have to be performed by a digital system enable the design engineer to determine the implementation of that system. System complexity and size are directly related to the complexity of the corresponding logic expressions and equations. Therefore, any algebraic simplification will lead to minimization of system complexity and size. Simplifying logic equations by applying the appropriate theorems is possible, but it often becomes a cumbersome and time-consuming chore. Therefore, special minimization techniques have been developed to obtain the desired result faster and more accurately. Most common, for a small number of variables, is the use of Karnaugh maps.

A Karnaugh map is a graphic representation of *all unique combinations* of all variables involved. Thus, a Karnaugh map may be considered to be the graphic representation of the minterm canonical form. Each minterm is represented by a cell, and the cells are assembled in an orderly arrangement such that adjacent cells represent minterms which differ by *only one variable.* A Karnaugh map for four variables is shown in Fig. 6.3. The diagonal line at the upper left-hand corner of the matrix indicates that the variables A and B are represented by the binary notations across the top of the matrix in the same sequence (right-hand digit refers to A and left-hand digit to B). The combinations of A and B as indicated by the binary notations on top are contained in the column below each assignment. The variables C and D are assigned down the side of the matrix and are contained in the horizontal rows of the matrix.

The minterm represented by a cell is determined by the binary assignments of the variables for that cell. A 0 denotes a *complemented* variable, and a 1 an *uncomplemented* variable. A decimal numerical value is assigned to each cell with variable A as the least significant bit.

From Fig. 6.3 it can be seen that the cell arrangement, because of the special sequence of the binary notations, is such that minterms of adjacent cells are identical except for one variable which appears complemented in one cell and uncomplemented in the adjacent cell. According to this definition of adjacency, the cells at the extreme ends of the same horizontal row are also to be considered adjacent. The same applies to the top and bottom cells of a column. As a result, the four corner cells of a Karnaugh map also must be considered to be adjacent.

A logic expression is mapped by entering a 1 in the cells representing minterms that are contained in the expression, whereas a 0 is entered in the cells representing nonpresent minterms. By applying the distributive property [$AB + AC = A(B + C)$] and the theorems 5*a* ($A + \bar{A} = 1$) and 1*b* ($A \cdot 1 = A$) to the minterms of two adjacent cells, one variable can be eliminated, because the condition or operation determined by the two minterms is independent of the eliminated variable. The two combined cells may again be combined with two other adjacent cells if the two cell groups are adjacent to each other. Thus, a group of four adjacent

$DC \backslash BA$	00	01	11	10
00	0 $\bar{D}\bar{C}\bar{B}\bar{A}$	1 $\bar{D}\bar{C}\bar{B}A$	3 $\bar{D}\bar{C}BA$	2 $\bar{D}\bar{C}B\bar{A}$
01	4 $\bar{D}C\bar{B}\bar{A}$	5 $\bar{D}C\bar{B}A$	7 $\bar{D}CBA$	6 $\bar{D}CB\bar{A}$
11	12 $DC\bar{B}\bar{A}$	13 $DC\bar{B}A$	15 $DCBA$	14 $DCB\bar{A}$
10	8 $D\bar{C}\bar{B}\bar{A}$	9 $D\bar{C}\bar{B}A$	11 $D\bar{C}BA$	10 $D\bar{C}B\bar{A}$

Fig. 6.3. Four-variable Karnaugh map with minterm designation and numerical assignment.

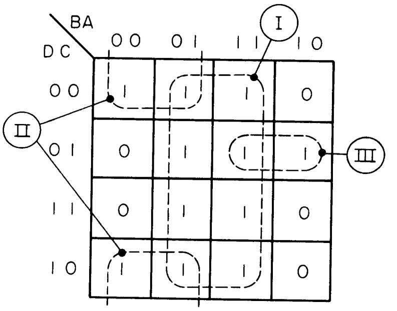


Fig. 6.4. Karnaugh map of Y.

cells can be combined, resulting in the elimination of two variables. This process of combining adjacent groups of combined cells may be continued as far as possible.

In general, the combination of 2^N adjacent cells results in the elimination of N variables. This can also be stated as

$$n = \log_2 m$$

where n = number of cells eliminated; $n = 0, 1, 2, 3, \ldots$
m = number of cells combined; $m = 1, 2, 3, 8, 16, \ldots$

An example follows:

$$Y = ABC + AB\bar{D} + A\bar{B}C + A\bar{B}D + A\bar{C}D + \bar{A}\bar{B}\bar{C} + BC\bar{D} + \bar{B}\bar{C}\bar{D}$$

ABC contains the minterms: $ABCD + ABC\bar{D}$
$AB\bar{D}$ contains the minterms: $ABC\bar{D} + AB\bar{C}\bar{D}$
$A\bar{B}C$ contains the minterms: $A\bar{B}CD + A\bar{B}C\bar{D}$
$A\bar{B}D$ contains the minterms: $A\bar{B}CD + A\bar{B}\bar{C}D$
$A\bar{C}D$ contains the minterms: $AB\bar{C}D + A\bar{B}\bar{C}D$
$\bar{A}\bar{B}\bar{C}$ contains the minterms: $\bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$
$BC\bar{D}$ contains the minterms: $ABC\bar{D} + \bar{A}BC\bar{D}$
$\bar{B}\bar{C}\bar{D}$ contains the minterms: $A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D}$

The Karnaugh map of the function Y is shown in Fig. 6.4.

The highest order of simplification will be attained when the 1s in the Karnaugh map are combined in as *few groups* as possible, each of which contains as *many adjacent cells* as possible. It must be understood, however, that the number of adjacent cells in each group must be the result of an integral power of 2; otherwise theorem 5*a cannot be applied to all cells* of the group. Groups of combined adjacent cells may share one or more cells with one or more other groups. This is permissible because an OR-ing of identical variables, terms, or minterms results in the variable, term, or minterm itself, as is indicated by theorem 3*a* ($A + A = A$).

The 1s in the Karnaugh map of Fig. 6.4 can be combined into three groups (I, II, and III). It can be derived (using theorems 5*a* and 1*b*) that the terms represented by the three groups are:

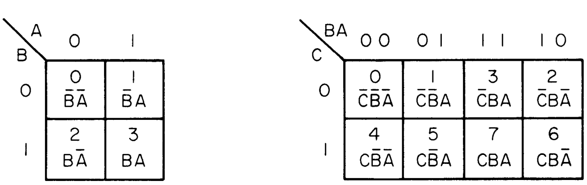
Group I: A
Group II: $\bar{B}\bar{C}$
Group III: $BC\bar{D}$

Thus,

$$\begin{aligned} Y &= ABC + AB\bar{D} + A\bar{B}C + A\bar{B}D + A\bar{C}D + \bar{A}\bar{B}\bar{C} + BC\bar{D} + \bar{B}C\bar{D} \\ &= A + \bar{B}\bar{C} + BC\bar{D} \end{aligned}$$

The terms represented by the three groups can be more easily found by examining the location of the three groups. Group I is located in two columns, and according to the binary notations of those columns, the term represented by group I is true for $A = 1$ (in both columns) no matter whether B is a 0 or a 1. This means that the term is independent of B, which in this case is a so-called *redundant variable.* Group I is also located in all four rows which contain all possible combinations of C and D. Thus, the term represented by group I is independent of C and D, which are consequently also redundant variables for group I. As a result, the term represented by group I is A. The location of group II is such that it represents a term which is true for $B = 0$ (in both columns) with A as a redundant variable, and is also true for $C = 0$ (both rows) with D as a redundant variable. Consequently, the term represented by group II is $\bar{B}\bar{C}$. The term represented by group III is true for $B = 1$ (both columns) with A as a redundant variable and for $C = 1$ and for $D = 0$. Thus, the term represented by group III is $BC\bar{D}$.

Since the logic expression of the preceding example contained four variables, a four-variable Karnaugh map was used. Figure 6.5 shows Karnaugh maps with minterm and decimal numerical values for two and three variables. The Karnaugh map is a valuable minimization tool in logic design. It is not very well suited, however, when a large (greater than 6) number of variables is involved, because the location of adjacent cells (in the sense of a one-variable difference) is in those cases not immediately obvious. A more suitable minimization technique for large numbers of variables is the Quinte-McCluskey method, which has the additional advantage of being easily programmed for solution by computer. This method will not be discussed here. Interested readers are referred to items 2, 3, and 4 of the Bibliography.



(a)

(b)

Fig. 6.5. Karnaugh maps: (*a*) for two variables; (*b*) for three variables.

6.5 PRACTICAL LOGIC GATES

It has been previously stated that the three basic logic functions are AND, OR, and NOT. Many logic gates, however, combine two or more of the basic logic functions. The most common logic gates are:

1. NAND GATE. The NAND gate is AND followed by a NOT operation. This is indicated by a circle at the gate output. The output Y is at a logical 0 if and only if all its inputs are at a logical 1. The symbol, logic expression, and truth table for a 2-input NAND gate are shown in Fig. 6.6*a*.
2. NOR GATE. The NOR gate is an OR followed by a NOT operation (circle at the gate output). The output Y is at a logical 0. The symbol, logic expression, and truth table for a NOR gate with two inputs (variables) are given in Fig. 6.6*b*.
3. NOT (INVERTER). The inverter which performs the NOT operation (complementing operation) is shown in Fig. 6.6*c*.

The logic expression for the output signal of a 2-input NAND gate (see Fig. 6.6*a*) is

$$Y = \overline{AB}$$

According to De Morgan's theorem, this may be written as

$$A = \overline{AB} = \bar{A} + \bar{B}$$

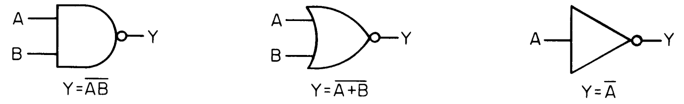
From this expression it can be seen that a single NAND gate performs an OR operation on its *complemented* input signals. Thus, a gate performing the NAND operation can be represented by either of the two symbols in Fig. 6.7.

For a 2-input NOR gate the logic expression (see Fig. 6.6*b*) of the output signal is

$$Y = \overline{A + B}$$

This can also be written according to De Morgan's theorem as

$$Y = \overline{A + B} = \bar{A} \cdot \bar{B}$$



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(a) NAND gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

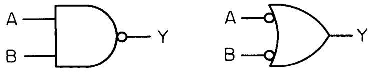
(b) NOR gate

A	Y
0	1
1	0

(c) Inverter

Fig. 6.6. Logic gates: (*a*) NAND gate; (*b*) NOR gate; (*c*) inverter.

Fig. 6.7. Symbols for gates performing the NAND operation.



This means that a single NOR gate performs an AND operation on its *complemented* input signals. Consequently, gates performing the NOR operation may be represented by either of the two symbols of Fig. 6.8.

6.6 ANALYSIS OF LOGIC CIRCUITS

It is obvious that a variable appears complemented in the expression for the output signal if it has passed through an odd number of inversion levels and that it appears uncomplemented if the number of inversion levels is even. The consequences of De Morgan's theorem for NAND and NOR gates can be generalized as follows:

A NAND (NOR) gate performs an *OR* (*AND*) operation on its *complemented* input variables if it is at an *odd* number of inversion levels; whereas an *AND* (*OR*) operation will be performed on its *uncomplemented* input variables if it is at an *even* number of inversion levels.

The foregoing statements lead to the following set of rules for obtaining the logic expression of the output signal of an interconnected array of *inverting gates:*

1. Consider the gate from which the output signal will be obtained as the first (odd) level of inversion, the preceding gates as the second (even) level, etc.
2. Consider all NAND gates in odd levels to perform the OR operation.
3. Consider all NAND gates in even levels to perform the AND operation.
4. Consider all NOR gates in odd levels to perform the AND operation.
5. Consider all NOR gates in even levels to perform the OR operation.
6. All input variables entering gates in *odd* levels should appear *complemented* in the logic expression of the output signal.
7. All input variables entering gates in *even* levels should appear *uncomplemented* in the logic expression of the output signal.

Example (see Fig. 6.9)

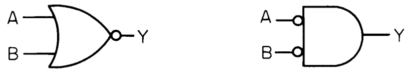
LOGIC EXPRESSION FOR Y_1. With regard to Y_1, the various inversion levels are located as follows:

a. NAND gate 5 is in the first (odd) level and must be considered to perform the OR operation.

b. NAND gate 4 is in the second (even) level and must be considered to perform the AND operation. Input variable F must appear uncomplemented in the expression for Y_1.

c. NOR gate 3 is in the second (even) level and must be considered to perform

Fig. 6.8. Symbols for gates performing the NOR operation.



the OR operation. Input variable C must appear uncomplemented in the expression for Y_1.

d. NOR gate 2 is in the third (odd) level and must be considered to perform the AND operation. Input variables D and E must appear complemented in the expression for Y_1.

e. NAND gate 1 is in the third (odd) level and must be considered to perform the OR operation. Input variables A and B must appear complemented in the expression for Y_1.

The logic expression for Y_1 is then:

$$Y_1 = \overbrace{[\underbrace{F \cdot}_{\substack{\text{Second (even)}\\ \text{inversion level}\\ \text{NAND gate 4}}} (\underbrace{\bar{D} \cdot \bar{E}}_{\substack{\text{Third (odd)}\\ \text{inversion level}\\ \text{NOR gate 2}}})] + [\underbrace{C +}_{\substack{\text{Second (even)}\\ \text{inversion level}\\ \text{NOR gate 3}}} (\underbrace{\bar{A} + \bar{B}}_{\substack{\text{Third (odd)}\\ \text{inversion level}\\ \text{NAND gate 1}}})]}^{\substack{\text{First odd}\\ \text{inversion level}\\ \text{NAND gate 5}}}$$

Thus,

$$Y_1 = \bar{A} + \bar{B} + C + \bar{D}\bar{E}F$$

LOGIC EXPRESSION FOR Y_2. For Y_2, the inversion levels have the following locations:

a. NOR gate 6: first (odd) level, AND operation, input variable G must appear complemented.

b. NAND gate 5: second (even) level, AND operation.

c. NAND gate 4: third (odd) level, OR operation, input variable F must appear complemented.

d. NOR gate 3: third (odd) level, AND operation, input variable C must appear complemented.

e. NOR gate 2: fourth (even) level, OR operation, input variables D and E must appear uncomplemented.

f. NAND gate 1: fourth (even) level, AND operation, input variables A and B must appear uncomplemented.

The logic expression for Y_2 is then

$$Y_2 = G \cdot \{[\bar{F} + (D + E)] \cdot [C \cdot (A \cdot B)]\} = ABC\bar{C}\bar{G}(D + E + \bar{F})$$

The expression for Y_2 can be obtained also in the following way:

$$Y_2 = \bar{Y}_1 \cdot \bar{G}$$

$$\bar{Y} = AB\bar{C}(D + E + \bar{F}) \qquad \text{rule of duality}$$

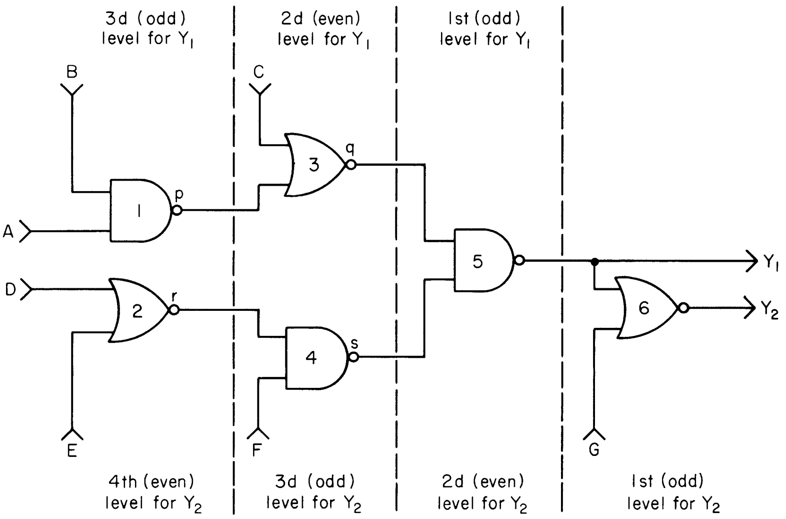

3d (odd) level for Y_1
2d (even) level for Y_1
Ist (odd) level for Y_1
B
C
q
3
p
1
A
5
Y_1
D
r
2
6
Y_2
s
4
E
F
G
4th (even) level for Y_2
3d (odd) level for Y_2
2d (even) level for Y_2
Ist (odd) level for Y_2

Fig. 6.9. Interconnected array of inverting gates.

Thus,

$$Y_2 = AB\bar{C}\bar{G}(D + E + \bar{F})$$

For more complicated arrays of interconnected inverting gates, the following method may be followed (see Fig. 6.9). Determine, as above, the location of the inversion levels and the logic operations of the various gates.

The logic expression of Y_1 can be written as

$$Y_1 = \bar{q} + \bar{s}$$

$$q = \bar{c} \cdot \bar{p} \qquad s = \bar{F} + \bar{r}$$

$$p = \bar{A} + \bar{B} \qquad r = \bar{D} \cdot \bar{E}$$

$$q = C \cdot \overline{(\bar{A} + \bar{B})} \qquad s = \bar{F} + \overline{(\bar{D} \cdot \bar{E})}$$

$$= \bar{C} \cdot A \cdot B \qquad = \bar{F} + D + E$$

Y_1 is then

$$Y_1 = \overline{(\bar{C} \cdot A \cdot B)} + \overline{(\bar{F} + D + E)} = \bar{A} + \bar{B} + C + \bar{D}\bar{E}F$$

The rules and methods outlined in the foregoing are valid for combinational logic consisting of *inverting* gates only. However, in a system with inverting (NAND and NOR) and noninverting (AND and OR) gates, the same rules and methods can be applied; but for the purpose of analysis, the noninverting gates are replaced by the corresponding inverting gates followed by an inverter. Because an inverter has only one input, it may be treated in the analysis as either a NAND or a NOR gate.

For example, the AND-OR-INVERT circuit of Fig. 6.10*a* may be represented by the circuit of Fig. 6.10*b*. Analyzing the circuit of Fig. 6.10*b* with the given rules

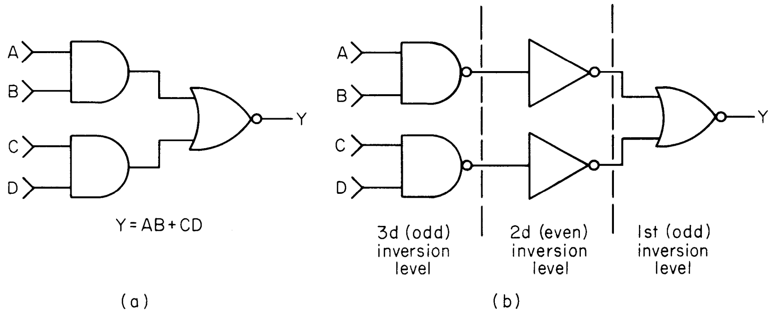

A
B
C
D
Y
Y=AB+CD
(a)
A
B
C
D
Y
3d (odd) inversion level
2d (even) inversion level
Ist (odd) inversion level
(b)

Fig. 6.10. Two circuits performing the AND-OR-INVERT operation.

results in

$$Y = (\bar{A} + \bar{B})(\bar{C} + \bar{D})$$

Application of De Morgan's theorem gives

$$Y = \overline{\overline{(\bar{A} + \bar{B})} + \overline{(\bar{C} + \bar{D})}} = \overline{AB + CD}$$

6.7 IMPLEMENTATION OF LOGIC EXPRESSIONS

The rules given for analyzing logic circuits are useful also for determining how a logic expression may be implemented. In general, a logic expression can be implemented in various ways. The implementation selected depends on considerations concerning propagation delay, number of gates, type of gates, signals already available in other parts of the system, etc. For example, implementations for the logic expression

$$Y = AB + AC + BC$$

may be derived as follows.

METHOD I. The expression consists of an OR-ing of three terms. Since a NAND gate in the first (odd) level performs an OR function, a 3-input NAND gate is

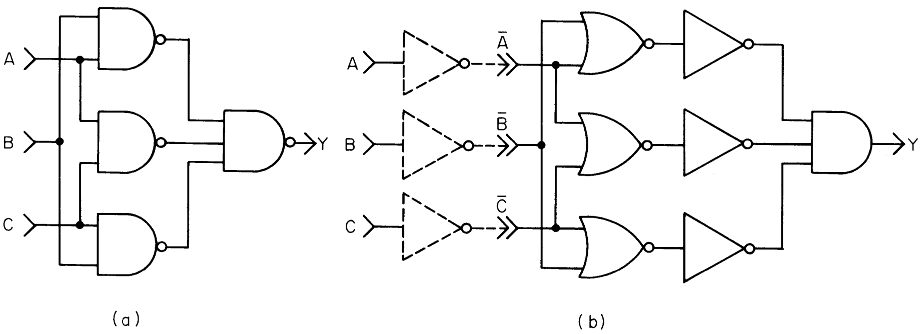

A
B
C
Y
(a)
A
B
C
$\bar{A}$
$\bar{B}$
$\bar{C}$
Y
(b)

Fig. 6.11. Two implementations for $Y = AB + AC + BC$.

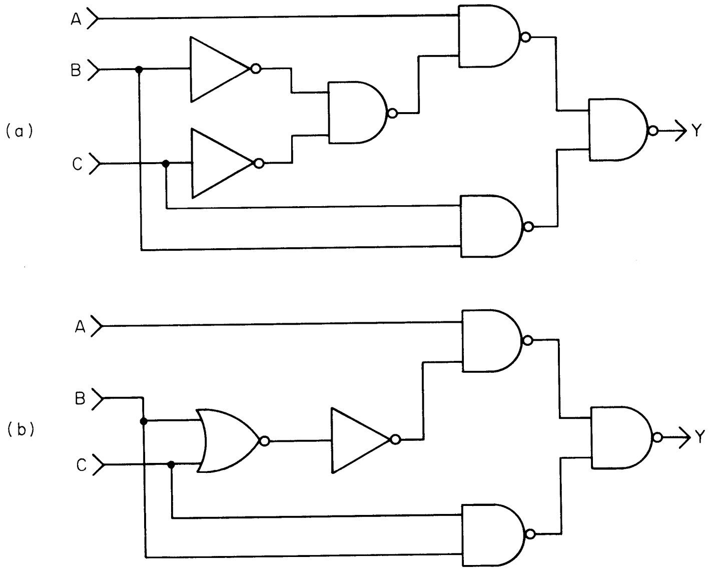


Fig. 6.12. Two implementations for $Y = A(B + C) + BC$.

necessary in the first level. The three OR-ed terms are AND-ings of two input variables. AND operations are performed by NAND gates in even levels or by NOR gates in odd levels. Consequently, at least two implementations are possible:

1. A 3-input NAND gate in the first level preceded by three 2-input NAND gates in the second level. This implementation is shown in Fig. 6.11*a*.
2. A 3-input NAND gate in the first level and three 2-input NOR gates in the third level (Fig. 6.11*b*). It is necessary to put three INVERTERS in the second level, because NOR gates in odd levels perform the OR operation on the inverted input variables. Inverters must be used unless complemented signals are available elsewhere in the system.

METHOD II. The desired logic expression can also be written as

$$Y = AB + AC + BC = A(B + C) + BC$$

The modified expression now contains an OR-ing of two terms, so that a 2-input NAND gate is necessary in the first level. To implement term BC, a 2-input NAND gate is needed in the second level. Another 2-input NAND gate in the second level is necessary to AND input variable A with $(B + C)$. The OR-ing $(B + C)$ must be implemented with a 2-input NAND gate in the third level or with a 3-input NOR gate in the fourth level. Both implementations are shown in Fig. 6.12.

6.8 IMPLEMENTATION OF FUNDAMENTAL LOGIC FUNCTIONS

NAND Circuits. In Series 54/74 integrated circuits, NAND gates with two, three, four, and eight inputs are available. NAND circuits with more than eight inputs

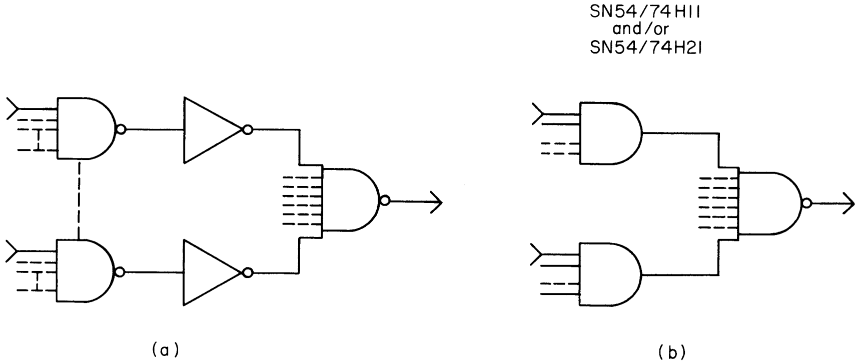


Fig. 6.13. NAND circuits with an expanded number of inputs.

may be formed by combining several gates. A brute-force method is shown in Fig. 6.13*a*, where the number of inputs of a NAND gate is expanded by means of AND circuits made of NAND gates followed by inverters. Using this type of implementation, NAND circuits with up to 64 inputs are possible.

Another solution is to use the AND gates such as the SN54/7408 (two inputs), SN54/74H11 (three inputs), or SN54/74H21 (four inputs) to expand the number of inputs of a NAND gate as shown in Fig. 6.13*b*. In this way, NAND circuits with up to 32 inputs may be achieved.

NOR Circuits. A quadruple 2-input NOR gate SN54/7402 is available in Series 54/74. This gate may be combined as in Fig. 6.14 to obtain a 4-input NOR circuit. Also available are AND-OR-INVERT gates which may be used as NOR circuits; in this case only, one input of each AND gate can be used for signal input. All other inputs of the AND input gates must be connected either to the signal input or to a logical 1 level.

Figure 6.15*a* shows an expandable AND-OR-INVERT gate (SN54/7450 or SN54/74H50) with expanders (SN54/7460 or SN54/74H60) in NOR circuit configuration for up to six inputs, and Fig. 6.15*b* shows the same technique for up to eight inputs.

AND Circuits. In the standard Series 54/74, there are two types of AND gates available: a quadruple 2-input AND gate with totem-pole output SN54/7408, and a quadruple 2-input AND gate with open-collector output SN54/7409. Two other types of AND gates are available in the high-speed TTL Series 54/74H: a triple 3-input AND gate SN54H/74H11 and a dual 4-input AND gate SN54H/74H21. These AND circuits may be used also to expand the number of inputs of any NAND

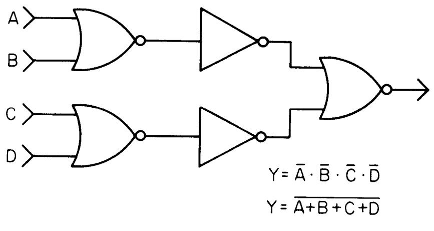


Fig. 6.14. 4-input NOR circuit.

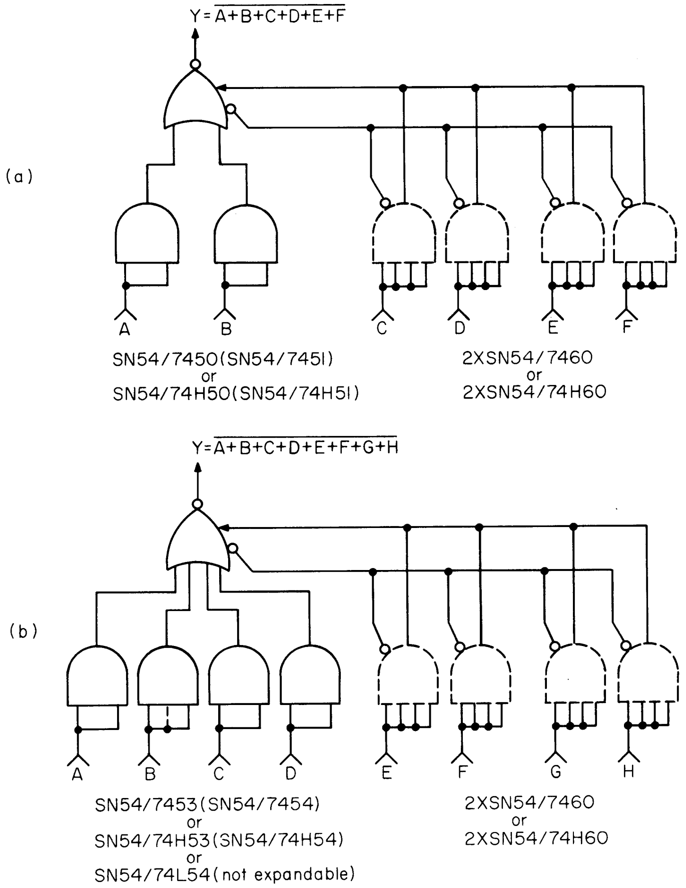

(a)
Y = $\overline{A+B+C+D+E+F}$
A
B
C
D
E
F
SN54/7450(SN54/7451)
or
SN54/74H50(SN54/74H51)
2XSN54/7460
or
2XSN54/74H60
(b)
Y=$\overline{A+B+C+D+E+F+G+H}$
A
B
C
D
E
F
G
H
SN54/7453(SN54/7454)
or
SN54/74H53(SN54/74H54)
or
SN54/74L54(not expandable)
2XSN54/7460
or
2XSN54/74H60

Fig. 6.15. Expandable AND-OR-INVERT gates in NOR circuit configuration.

and/or AND (input) gate. Increasing the number of inputs to a NAND circuit with high-speed series AND gates as expanders has been shown in Fig. 6.13*b*. The same can be done for AND circuits, as is shown in Fig. 6.16*a*. Using this configuration, AND circuits with up to 16 inputs can be implemented. It is also possible to make 2-, 3-, 4-, and 8-input AND circuits by inverting the output of NAND gates. By

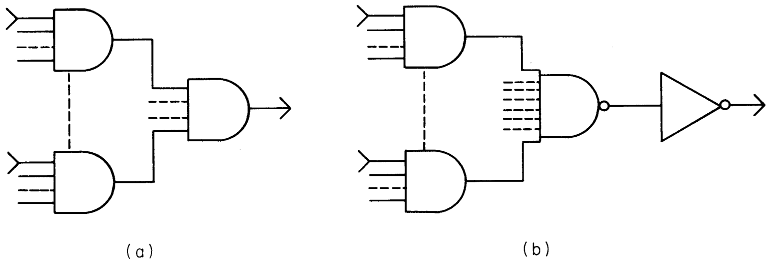


Fig. 6.16. AND circuits with AND and/or NAND gates.

using the high-speed series AND gates as expanders (Fig. 6.16*b*), AND circuits with up to 32 inputs are possible.

Combinational gate networks with a NOR gate SN54/7402 in the first inversion level and NAND gates in the second level (Fig. 6.17*a*) also perform the AND operation. In this way, AND circuits with up to 16 inputs are possible.

Instead of the SN54/7402 NOR gate, the NOR circuits of Fig. 6.15 may be used. With the NOR circuit of Fig. 6.15*a*, AND circuits with up to 48 inputs (Fig. 6.17*b*) can be made. AND circuits with up to 64 inputs are possible (Fig. 6.17*c*) with the NOR circuit of Fig. 6.15*b*.

As explained earlier, a NOR gate in an odd inversion level performs an AND operation on its complemented input variables. Such a circuit with the SN54/7402 NOR gate performing the AND operation on the variables A and B is shown in

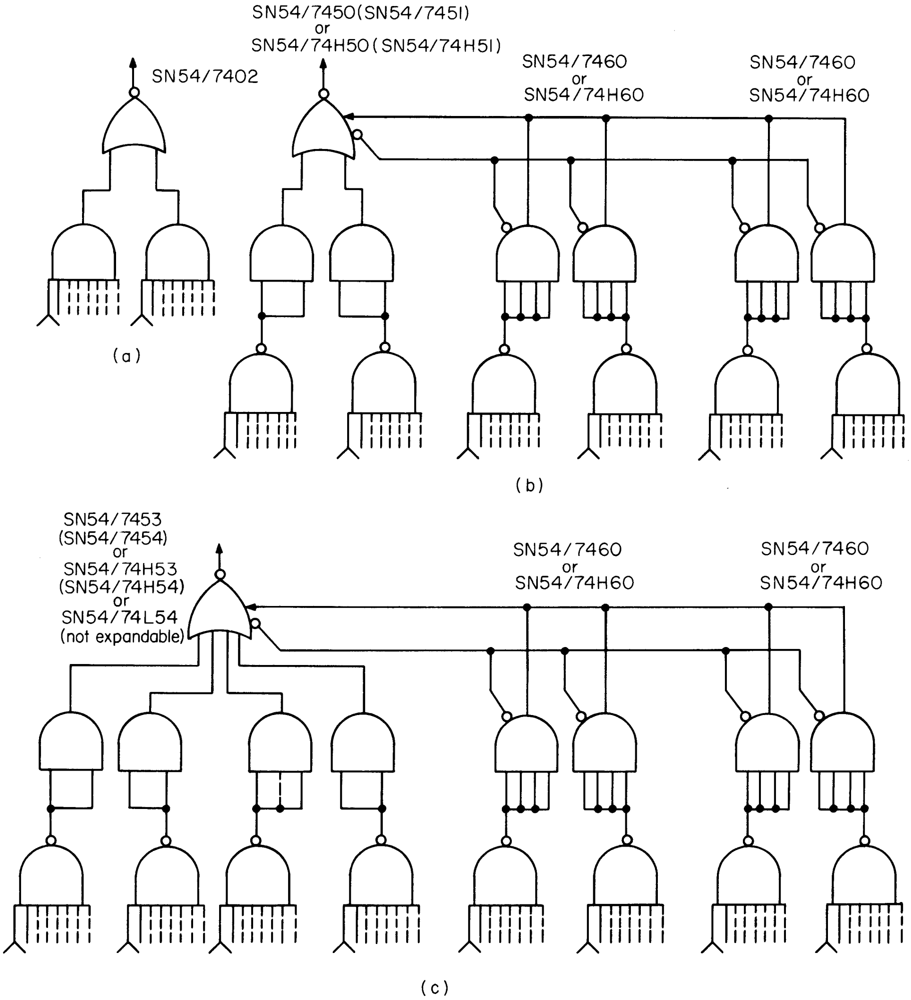

SN54/7402
(a)
SN54/7450(SN54/7451)
or
SN54/74H50(SN54/74H51)
SN54/7460
or
SN54/74H60
SN54/7460
or
SN54/74H60
(b)
SN54/7453
(SN54/7454)
or
SN54/74H53
(SN54/74H54)
or
SN54/74L54
(not expandable)
SN54/7460
or
SN54/74H60
SN54/7460
or
SN54/74H60
(c)

Fig. 6.17. Combinational AND circuits.

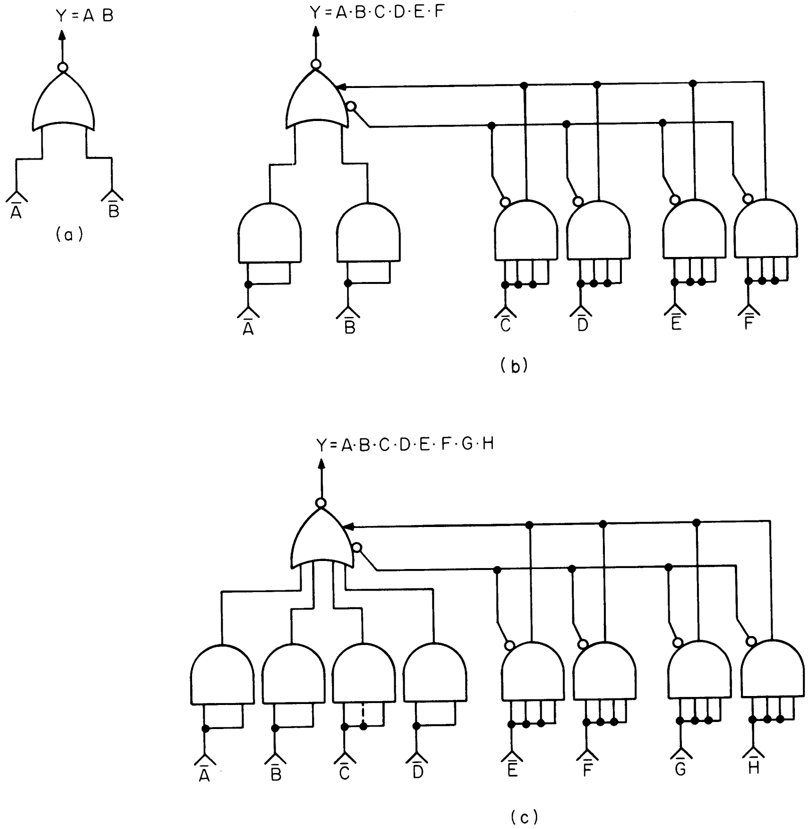


Fig. 6.18. AND operations with NOR circuits.

Fig. 6.18*a*. The NOR circuits of Fig. 6.15 may also be used in this fashion. With the NOR circuit of Fig. 6.15*a*, an AND circuit with up to six inputs (Fig. 6.18*b*) can be made, and an AND circuit with up to eight inputs (Fig. 6.18*c*) is possible, using the NOR circuit of Fig. 6.15*b*.

OR Circuits. Regular OR gates are not available in the standard Series SN54/74, but there is an expandable AND-OR gate (SN54/74H52) in the high-speed series. This gate may be used to perform the OR function. In this case, only one input of each AND gate is used for signal input. All other inputs of an AND input gate must be connected either to the signal input or to a logical 1 level. With the use of expanders (SN54/74H61) an OR gate with up to 10 inputs (Fig. 6.19) can be obtained.

It is also possible to make an OR circuit by inverting the output of the SN54/7402 2-input NOR gate. An OR circuit with up to six inputs can be obtained by inverting the output of the NOR circuit of Fig. 6.15*a*. Inversion of the output of the NOR circuit of Fig. 6.15*b* results in an OR circuit with up to eight inputs.

Combinational circuits with a NAND gate in the first level and NOR gates SN54/7402 in the second level (Fig. 6.20) also perform the OR operation. In this

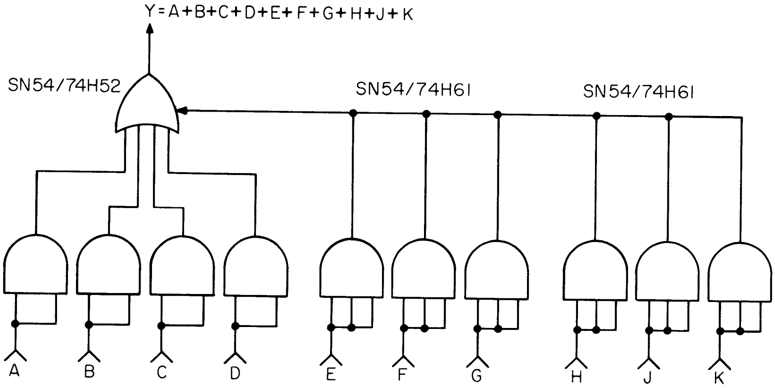


Fig. 6.19. OR circuit with AND-OR gate SN54/74H52 and expanders SN54/74H61.

way, OR circuits with up to 16 inputs may be obtained. With NOR circuits of the type in Fig. 6.15*a* in the second level and an 8-input NAND gate in the first level, an OR circuit with up to 48 inputs is possible. An OR circuit with 64 inputs can be obtained if NOR circuits of the type in Fig. 6.15*b* are used in the second level.

It was previously stated that a NAND gate in an odd level performs an OR operation on its complemented input variables. With the available NAND gates, OR operations can be performed on two, three, four, or eight variables, as shown in Fig. 6.21*a*. By expanding the number of inputs of the available NAND gates with AND gates of the high-speed series, OR operations can be performed on up to 32 variables, as shown in Fig. 6.21*b*.

Inverters. An inverter performs the NOT operation on one input variable only. In Series 54/74 several hex inverters (six inverters in one package) are available. The hex inverters of the high-speed series are SN54/74H04 with totem-pole output and SN54/74H05 with open-collector output. Both types are offered also in the standard series: SN54/7404 with totem-pole output and SN54/7405 with open-collector output. In the low-power series only one type of inverter is available, the SN54/74L04 with totem-pole output.

Every inverting gate can be used as an inverter either by connecting all inputs together or by using one input as signal input *and* connecting all other inputs to the appropriate *fixed* logical voltage levels as shown in Fig. 6.22.

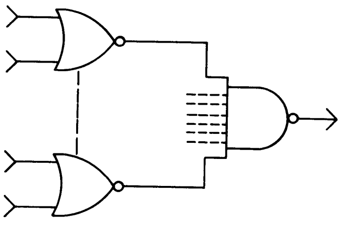


Fig. 6.20. Combinational OR circuit.

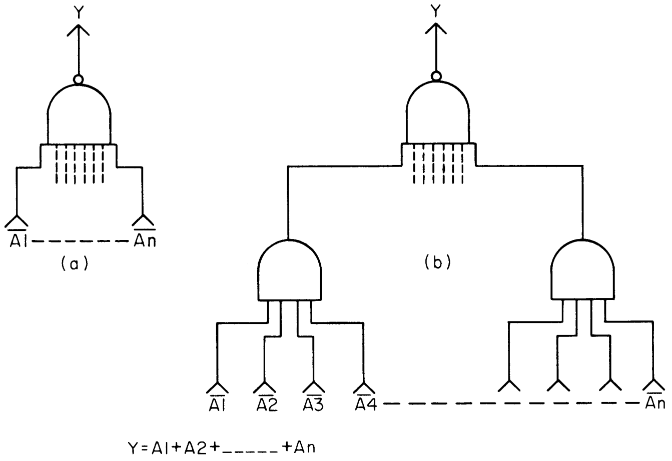

Y
Y
$\overline{A1}$ — — — — — — $\overline{An}$
(a)
(b)
$\overline{A1}$ $\overline{A2}$ $\overline{A3}$ $\overline{A4}$ — — — — — — — — — — — — — $\overline{An}$
Y = A1 + A2 + _ _ _ _ _ + An

Fig. 6.21. OR operations with NAND circuits.

AND-OR-INVERT Gates. A wide variety of AND-OR-INVERT (AOI) gates is available in the TTL Series 54/74. The logic diagram of a 2-wide 2-input version is shown in Fig. 6.23*a*. For the purpose of applying the rules for analyzing combinational arrays of inverting gates, the representation of Fig. 6.23*b* may be used.

From the logic expression of Fig. 6.23, it can be seen that an AOI gate performs an OR-AND operation on its *complemented* input variables. An AND-OR operation may be obtained by inverting the output signal of an AOI gate. Several types of AOI gates are expandable. Of those circuits, the OR branches can be expanded. For each expandable type, a maximum number of additional OR branches is specified in the data sheets.

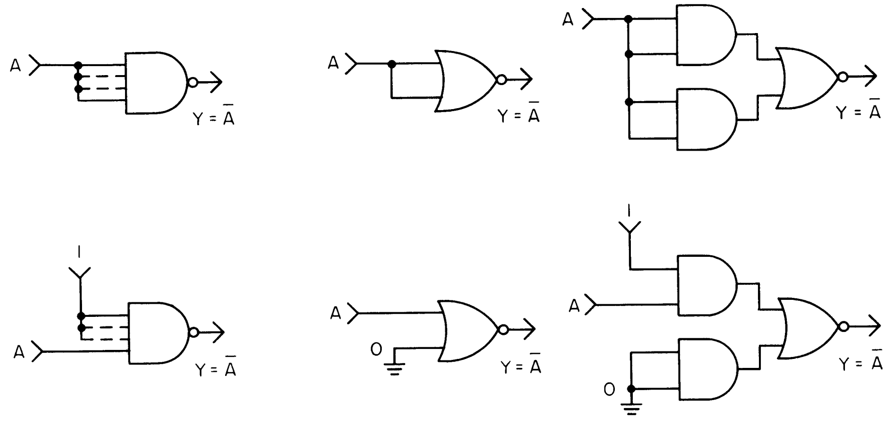
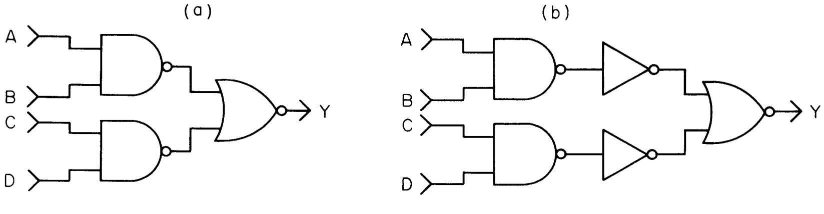

A
Y = $\overline{A}$
A
Y = $\overline{A}$
A
Y = $\overline{A}$
I
A
Y = $\overline{A}$
A
0
Y = $\overline{A}$
I
A
0
Y = $\overline{A}$

Fig. 6.22. Inverting gates used as inverters.



$$Y = \overline{AB + CD} = (\bar{A} + \bar{B}) \cdot (\bar{C} + \bar{D})$$

Fig. 6.23. Two representations of a 2-wide 2-input AND-OR-INVERT gate.

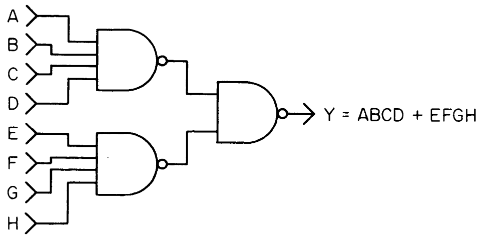

A
B
C
D
E
F
G
H
Y = ABCD + EFGH

Fig. 6.24. AND-OR circuit with NAND gates.

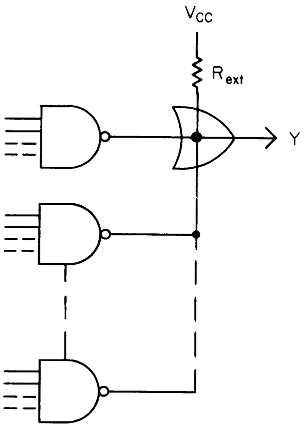
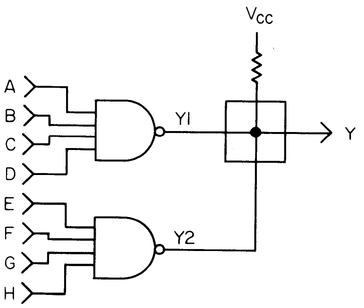


Fig. 6.25. Commonly used representation of collector-dotted NAND gates.


V_{CC}
A
B
C
D
E
F
G
H
Y1
Y2
Y

Truth table

$Y1 = \overline{ABCD}$	$Y2 = \overline{EFGH}$	Y
0	0	0
0	1	0
1	0	0
1	1	1

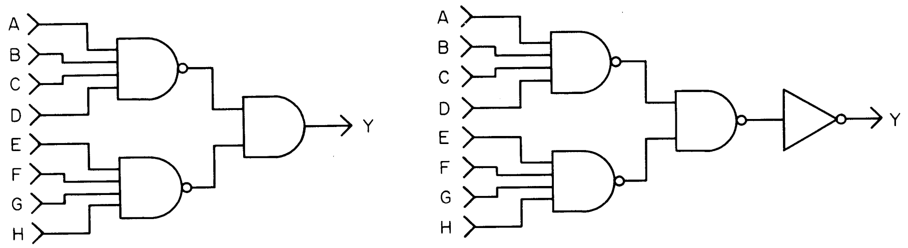
Fig. 6.26. Logic operation performed by collector-dotting.

AND-OR Circuits. In the high-speed Series 54/74H, a 4-wide expandable AND-OR gate is available. The number of OR branches may be expanded with a maximum of six additional OR branches to a total of ten. As mentioned before, an AND-OR circuit may be obtained also by inverting the output signal of an AND-OR-INVERT gate. Application of the rules for analyzing and synthesizing combinational arrays of inverting gates to an AND-OR expression such as $Y = ABCD + EFGH$ results in an AND-OR circuit, as is shown in Fig. 6.24. This circuit consists of a NAND gate in the first inversion level (OR function) and NAND gates in the second inversion level (AND functions).

Open-collector Gates. Several types of open-collector gates are available in Series 54/74. For a listing of open-collector gates refer to Chapter 3, Table 3.1. Open-collector gates must be used with external output pull-up resistors (collector resistors). This opens the possibility of connecting several open-collector outputs together in combination with a single pull-up resistor. Such a configuration performs a logic operation which is often referred to as *wired-OR logic, dot-OR logic,* or *collector-dotting,* and is symbolized as in Fig. 6.25. Although widely used, the terms "wired-OR" and "dot-OR," and the OR symbol at the output node of Fig. 6.25 are misleading.

Examination of the output signals of the various open-collector gates and the resulting output signal of the whole collector-dotted array reveals that the latter is a logical 1 *if, and only if,* the output signals of *all* collector-dotted gates are logical 1s. The resulting output signal becomes a logical 0 as soon as one or more outputs become a logical 0. This is shown in Fig. 6.26 for two collector-dotted 4-input NAND gates. The truth table of Fig. 6.26 is that of an AND function, and it shows that collector-dotting performs an AND operation on the output signals of collector-dotted gates. The circuit of Fig. 6.26 may thus be represented by the circuits shown in Fig. 6.27.

The expression for the output signal derived from Fig. 6.27*b* shows that collector-dotted NAND gates perform an OR-AND operation on the complemented input variables. According to De Morgan's theorem, the expressions of Fig. 6.27 may



$Y = \overline{ABCD} \cdot \overline{EFGH}$

(a)

$Y = (\bar{A} + \bar{B} + \bar{C} + \bar{D}) \cdot (\bar{E} + \bar{F} + \bar{G} + \bar{H})$

(b)

Fig. 6.27. (*a*) Functional representation of collector-dotted NAND gates; (*b*) circuit representation for applying the rules for analyzing combinational arrays of inverting gates.

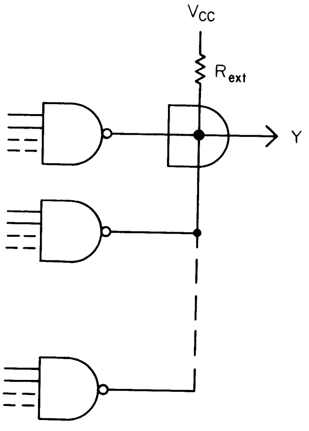


Fig. 6.28. Correct representation of collector-dotting.

be written as

$$Y = \overline{ABCD + EFGH}$$

or

$$\overline{Y} = ABCD + EFGH$$

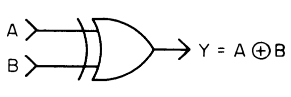
It is apparent from these expressions that collector-dotted NAND gates perform the same operation as AND-OR-INVERT gates; i.e., an inverted OR operation on the AND-ed input variable of the various NAND gates. This interpretation of the logic performance of collector-dotted gates is largely responsible for the misleading terms "wired-OR" and "dot-OR." Thus, the correct description of logic performed by collector-dotting is *wire-AND logic* and should be represented as in Fig. 6.28.

Exclusive-OR Circuits. An exclusive-OR function of two variables is characterized by the logic expression

$$Y = A \oplus B = (A + B)(\overline{A} + \overline{B}) = A\overline{B} + \overline{A}B$$

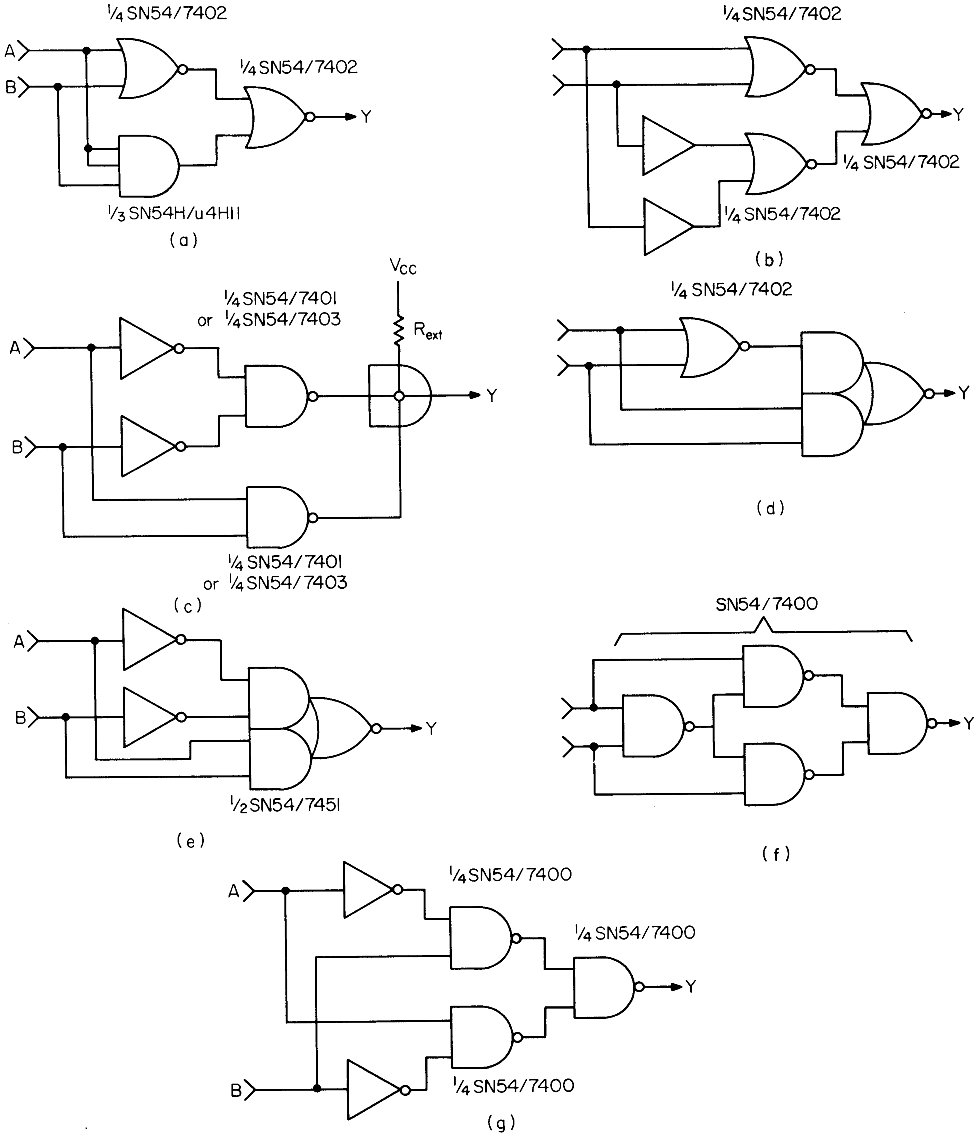
The logic symbol and the truth table for a 2-variable exclusive-OR circuit are shown in Fig. 6.29. The standard and low-power Series 54/74 include quadruple 2-input exclusive-OR gates SN54/7486 and SN54L/74L86. In special cases, where fewer than four circuits are necessary, and where the appropriate logic signals are already present, 2-input exclusive-OR circuits may be implemented as in Fig. 6.30.

According to the truth table of Fig. 6.29, a 2-input exclusive-OR circuit gives a logical 1 output signal whenever its input signals are *unequal* ($A = 1$, $B = 0$; or $A = 0$, $B = 1$). It should be noted that the same truth table applies when adding



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 6.29. Symbol and truth table for exclusive-OR circuit.


¼SN54/7402
A
B
¼SN54/7402
Y
⅓SN54H/u4H11
(a)
¼SN54/7402
Y
¼SN54/7402
¼SN54/7402
(b)
Vcc
¼SN54/7401
or ¼SN54/7403
R_{ext}
A
Y
B
¼SN54/7401
or ¼SN54/7403
(c)
¼SN54/7402
Y
(d)
A
B
Y
½SN54/7451
(e)
SN54/7400
Y
(f)
¼SN54/7400
A
¼SN54/7400
Y
B
¼SN54/7400
(g)

$$Y = A \oplus B = (A + B)(A + B) = AB + AB$$

Fig. 6.30. 2-input exclusive-OR circuits.

two *binary digits* (bits). A 2-input exclusive-OR circuit is therefore sometimes called a *modulo-2 adder* or a *half-adder*. The name half-adder refers to the fact that a possible carry-bit, resulting from an addition of two preceding bits, has not been taken into account. A full addition is performed by a second exclusive-OR circuit with the output signal of the first circuit and the carry as input signals, as shown in Fig. 6.31.

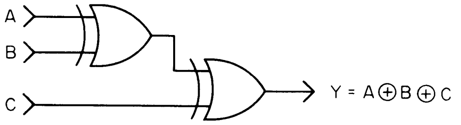


Fig. 6.31. Cascading of two exclusive-OR circuits.

The configuration of Fig. 6.31 is a cascading of two exclusive-OR circuits resulting in an exclusive-OR-ing of three variables *A*, *B*, and *C*. Consequently, the sum output of a full-adder for two bits is an exclusive-OR-ing of the two bits to be added and the carry of the preceding adding stage. The logic expression of an exclusive-OR-ing of three variables *A*, *B*, and *C* is

$$\begin{aligned} A \oplus B \oplus C &= (A\bar{B} + \bar{A}B)\bar{C} + \overline{(A\bar{B} + \bar{A}B)}\bar{C} \\ &= (A\bar{B} + \bar{A}B)C + (\bar{A}\bar{B} + AB)C \\ &= A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC \end{aligned}$$

In general, an exclusive-OR-ing of *n* variables results in a logical 1 output if an *odd* number of the input variables are 1s. An exclusive-OR-ing of *n* variables may be obtained by cascading 2-input exclusive-OR circuits. Two ways of cascading are possible, as shown in Fig. 6.32.

Both cascading methods require the same number of 2-input exclusive-OR circuits. With the method of Fig. 6.32*b*, however, the output signal is obtained much faster, in a maximum of three gate propagation delays. On the other hand, the

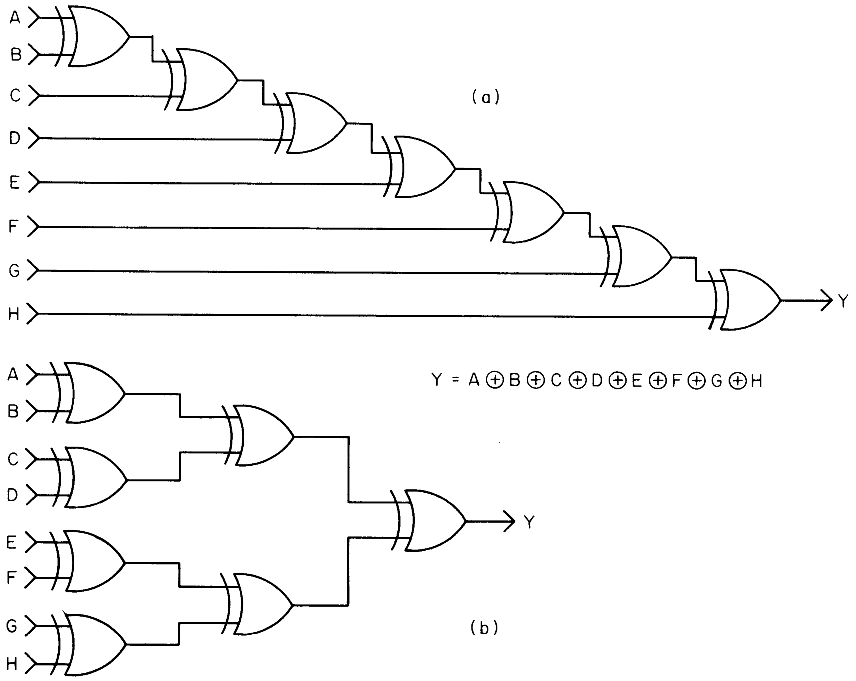


Fig. 6.32. Two ways of cascading 2-input exclusive-OR circuits.

Fig. 6.33. Truth table for a 2-input exclusive-NOR function.

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

method of Fig. 6.32*a* requires a maximum of seven gate propagation delays. For exclusive-OR operations, the following rules can be derived:

$$A \oplus A = A \cdot \overline{A} + \overline{A} \cdot A = 0$$

$$A \oplus \overline{A} = A \cdot A + \overline{A} \cdot \overline{A} = 1$$

$$A \oplus 1 = A \cdot 0 + \overline{A} \cdot 1 = \overline{A}$$

$$A \oplus 0 = A \cdot 1 + \overline{A} \cdot 0 = A$$

$$\begin{aligned} A \oplus A \oplus A \oplus \cdots \oplus A &= 0 \text{ if total number of terms is even} \\ &= A \text{ if total number of terms is odd} \\ \overline{A} \oplus \overline{A} \oplus \overline{A} \cdots \oplus \overline{A} &= 1 \text{ if total number of terms is even} \\ &= A \text{ if total number of terms is odd} \\ A \oplus 1 \oplus 1 \oplus \cdots \oplus 1 &= \overline{A} \text{ if total number of terms is even} \\ &= A \text{ if total number of terms is odd} \\ A \oplus 0 \oplus 0 \oplus \cdots \oplus 0 &= A \end{aligned}$$

Exclusive-NOR (Comparator) Circuits. Another widely used logic function of two variables is characterized by the expression

$$Y = (A + \overline{B})(\overline{A} + B) = AB + \overline{A}\,\overline{B}$$

A circuit performing this operation may be obtained by inverting the output signal of a 2-input exclusive-OR circuit. According to De Morgan's theorem,

$$\overline{A + \overline{\overline{B}}} = \overline{A\overline{B} + \overline{A}B} = (\overline{A} + B)(A + \overline{B}) = \overline{A}\,\overline{B} + AB$$

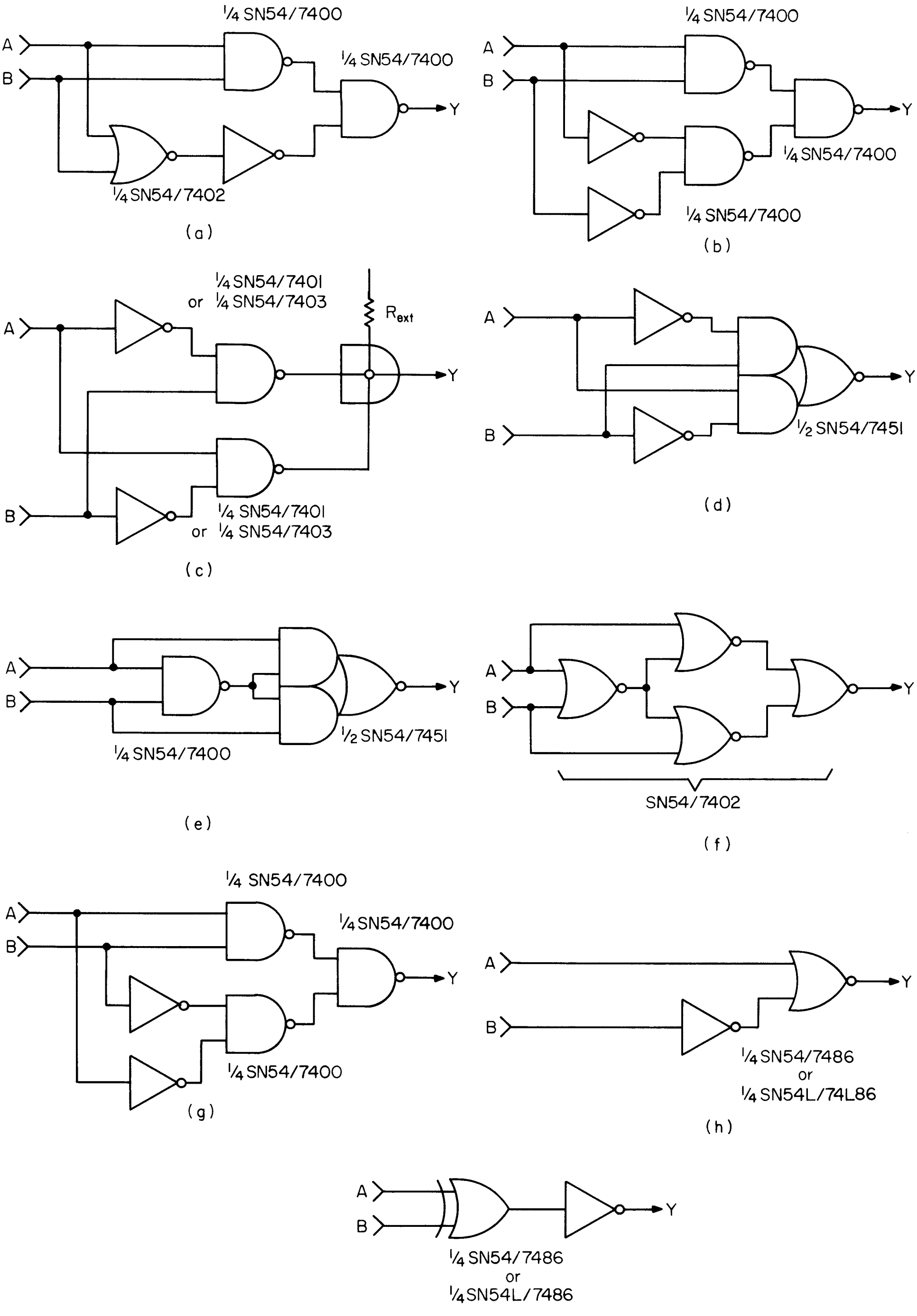
The truth table of this logic function is given in Fig. 6.33; it shows that a logical 1 output is obtained whenever the two input signals are equal ($A = 1$, $B = 1$; or $A = 0$, $B = 0$).

A circuit performing this operation is known under several names: *exclusive-NOR, comparator,* or *coincidence* circuit. Besides inverting the output signal of an exclusive-OR circuit, 2-input exclusive-NOR circuits may be implemented as in Fig. 6.34.

6.9 COMBINATIONAL LOGIC APPLICATIONS

Gray-to-binary Converter. The Gray code is a reflected digital code with the special property that two adjacent Gray code numbers differ *by only 1 bit.* For 4 bits, the binary code and the corresponding Gray code are shown in Fig. 6.35.

A combinational logic network is required that gives, for any Gray code number


¼ SN54/7400
A
B
¼ SN54/7400
Y
¼ SN54/7402
(a)
¼ SN54/7400
A
B
Y
¼ SN54/7400
¼ SN54/7400
(b)
¼ SN54/7401
or ¼ SN54/7403
R_ext
A
Y
B
¼ SN54/7401
or ¼ SN54/7403
(c)
A
Y
B
½ SN54/7451
(d)
A
B
Y
¼ SN54/7400
½ SN54/7451
(e)
A
B
Y
SN54/7402
(f)
¼ SN54/7400
A
B
¼ SN54/7400
Y
¼ SN54/7400
(g)
A
Y
B
¼ SN54/7486
or
¼ SN54L/74L86
(h)
A
B
Y
¼ SN54/7486
or
¼ SN54L/7486
(i)

$$Y = \overline{A \oplus B} = AB + \overline{A}\overline{B} = (A + \overline{B})(\overline{A} + B)$$

Fig. 6.34. 2-input exclusive-NOR circuits.

	Gray				Binary			
	G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	0	1	1
4	0	1	1	0	0	1	0	0
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	1	0
7	0	1	0	0	0	1	1	1
8	1	1	0	0	1	0	0	0
9	1	1	0	1	1	0	0	1
10	1	1	1	1	1	0	1	0
11	1	1	1	0	1	0	1	1
12	1	0	1	0	1	1	0	0
13	1	0	1	1	1	1	0	1
14	1	0	0	1	1	1	1	0
15	1	0	0	0	1	1	1	1

Figure 6.35

at its input, the corresponding binary number at its outputs. A convenient way to arrive at the logic expressions for B_0 through B_3, in terms of G_0 through G_3, is to use the Karnaugh maps of Fig. 6.36. Converting a 4-bit Gray code into a 4-bit binary code results in the following logic expressions:

$$B_0 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$$

$$B_1 = G_1 \oplus G_2 \oplus G_3$$

$$B_2 = G_2 \oplus G_2$$

$$B_3 = G_3$$

In general, $B_n = G_n \oplus G_{(n+1)} \oplus G_{(n+2)} \oplus G_{(n+3)} \oplus \cdots$, with the understanding that nonexistent bits are considered to be 0. The result is that the most significant bit (MSB) in the Gray code and in the binary code are identical, and that the less significant bits are obtained by a cascading of exclusive-OR operations on the Gray code bits. A logic diagram of a Gray-to-binary converter for n bits is shown in Fig. 6.37.

Binary-to-Gray Converter. The same procedure as in the foregoing can be used for a binary-to-Gray converter, only now the conversion table (Fig. 6.35) is to be used from right to left, with the binary code as input and the Gray code as the output. Figure 6.38 shows corresponding Karnaugh maps. Converting a 4-bit binary code into a 4-bit Gray code results in the following logic expressions:

$$G_0 = B_0 \oplus B_1$$

$$G_1 = B_1 \oplus B_2$$

$$G_2 = B_2 \oplus B_3$$

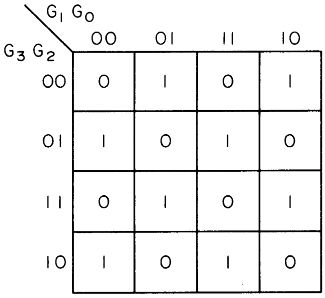
$$G_3 = B_3$$

In general, $G_n = B_n \oplus B_{(n+1)}$ again, with the understanding that nonexistent bits

Karnaugh map for B_0

$$B_0 = \bar{G}_2\bar{G}_3 (G_0\bar{G}_1 + \bar{G}_0 G_1) + G_2\bar{G}_3 (G_0 G_1 + \bar{G}_0\bar{G}_1)$$
$$+ G_2 G_3 (G_0\bar{G}_1 + \bar{G}_0 G_1) + \bar{G}_2 G_3 (G_0 G_1 + \bar{G}_0\bar{G}_1)$$
$$= (G_0\bar{G}_1 + \bar{G}_0 G_1)(G_2 G_3 + \bar{G}_2\bar{G}_3) + (G_0 G_1 + \bar{G}_0\bar{G}_1)(G_2\bar{G}_3 + \bar{G}_2 G_3)$$
$$= (G_0 \oplus G_1)(\overline{G_2 \oplus G_3}) + (\overline{G_0 \oplus G_1})(G_2 \oplus G_3)$$

$$B_0 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$$

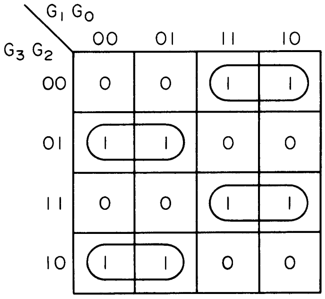

G₁ G₀
G₃ G₂
00 01 11 10
00 0 1 0 1
01 1 0 1 0
11 0 1 0 1
10 1 0 1 0

B_0

Karnaugh map for B_1

$$B_1 = G_1\bar{G}_2\bar{G}_3 + \bar{G}_1 G_2\bar{G}_3 + G_1 G_2 G_3 + \bar{G}_1\bar{G}_2 G_3$$
$$= G_1(G_2 G_3 + \bar{G}_2\bar{G}_3) + \bar{G}_1 (G_2\bar{G}_3 + \bar{G}_2 G_3)$$
$$= G_1(\overline{G_2 \oplus G_3}) + \bar{G}_1 (G_2 \oplus G_3)$$

$$B_1 = G_1 \oplus G_2 \oplus G_3$$

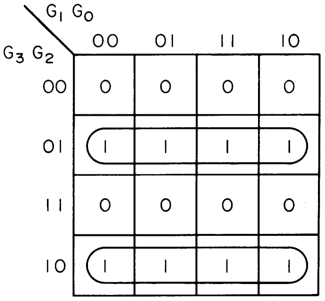

G₁ G₀
G₃ G₂
00 01 11 10
00 0 0 1 1
01 1 1 0 0
11 0 0 1 1
10 1 1 0 0

B_1

Karnaugh map for B_2

$$B_2 = G_2\bar{G}_3 + \bar{G}_2 G_3$$

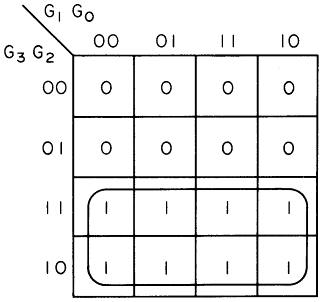
$$B_2 = G_2 \oplus G_3$$


G₁ G₀
G₃ G₂
00 01 11 10
00 0 0 0 0
01 1 1 1 1
11 0 0 0 0
10 1 1 1 1

B_2

Karnaugh map for B_3

$$B_3 = G_3$$


G₁ G₀
G₃ G₂
00 01 11 10
00 0 0 0 0
01 0 0 0 0
11 1 1 1 1
10 1 1 1 1

B_3

Figure 6.36

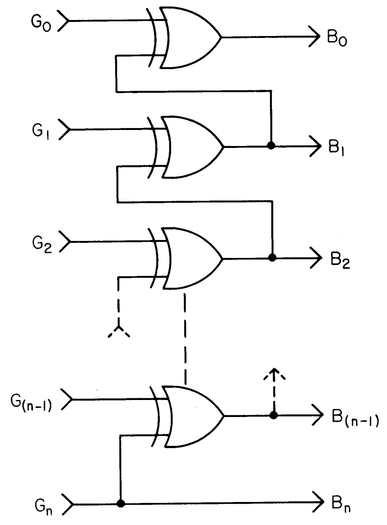


Fig. 6.37. Gray-to-binary converter for n bits.

are considered to be 0. A logic diagram of a binary-to-Gray converter for n bits is shown in Fig. 6.39.

Combined Binary-to-Gray and Gray-to-binary Converter. The two converters previously discussed may be combined to obtain a controllable dual-purpose converter as shown in Fig. 6.40.

Excess-3-to-BCD Converter. The excess-3 code is a *self-complementing* binary-coded decimal number system. Self-complementing means that the 1's complement of a number can be obtained by inverting each bit of that number. The excess-3 code and the corresponding binary code are given in Fig. 6.41. It should be noted that in the excess-3 code, the six states shown in Fig. 6.42 are never used. Consequently, these six states may be called "don't care" states, and the corresponding cells in the Karnaugh maps (Fig. 6.43) may be made 1s or 0s at the designer's choice for the best minimization. Conversion of the excess-3 code into BCD requires the following logic expressions:

$$A = \bar{E}_0$$

$$B = E_0 \oplus E_1$$

$$C = \bar{E}_0\bar{E}_2 + E_0E_1E_2 + E_0\bar{E}_1E_3$$

$$D = E_0E_1E_3 + E_2E_3$$

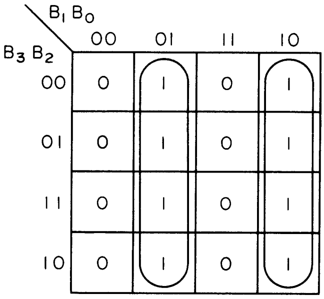
An implementation of an excess-3-to-BCD converter is shown in Fig. 6.44.

BCD-to-excess-3 Converter. For this conversion, Fig. 6.41 must be used from right to left. Notice that in the BCD code, the six states in Fig. 6.45 are not used. Thus, these six states are "don't care" states and are handled (see Fig. 6.46) as in the preceding example. An implementation of a BCD-to-excess-3 converter is shown in Fig. 6.47.

Karnaugh map for G_0

$G_0 = B_0\bar{B}_1 + \bar{B}_0B_1$

$G_0 = B_0 \oplus B_1$



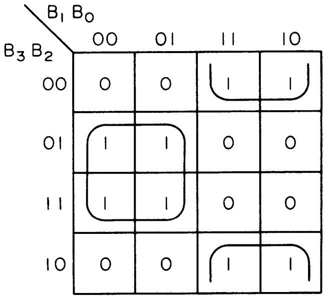
$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

G_0

Karnaugh map for G_1

$G_1 = B_1\bar{B}_2 + \bar{B}_1B_2$

$G_1 = B_1 \oplus B_2$



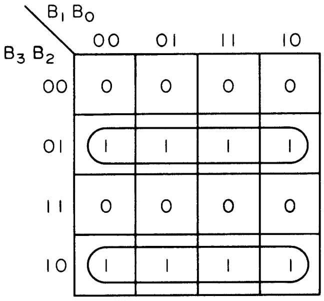
$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

G_1

Karnaugh map for G_2

$G_2 = B_2\bar{B}_3 + \bar{B}_2B_3$

$G_2 = B_2 \oplus B_3$

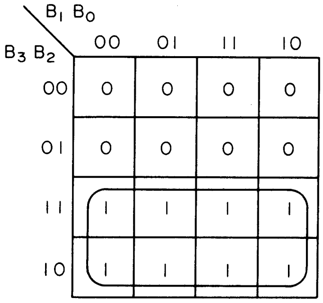


$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

G_2

Karnaugh map for G_3

$G_3 = B_3$



$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

G_3

Figure 6.38

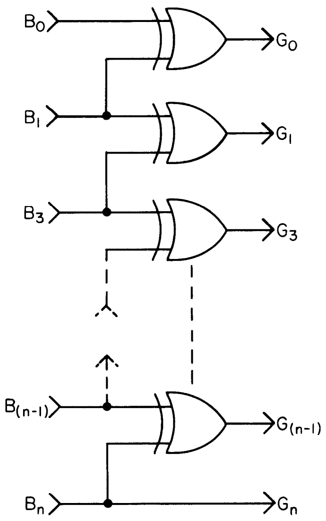

B0
B1
B3
B(n-1)
Bn
G0
G1
G3
G(n-1)
Gn

Fig. 6.39. Binary-to-Gray converter for *n* bits.

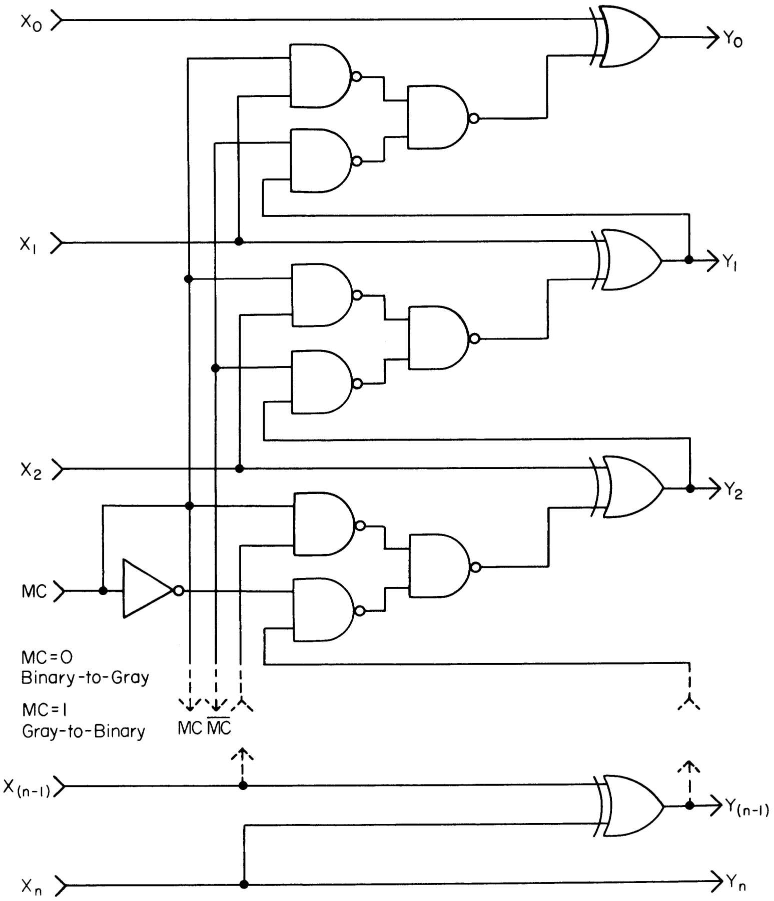

X0
X1
X2
MC
MC=0
Binary-to-Gray
MC=1
Gray-to-Binary
MC MC̄
X(n-1)
Xn
Y0
Y1
Y2
Y(n-1)
Yn

Fig. 6.40. Controllable binary-to-Gray/Gray-to-binary converter.

	Excess-3				BCD			
	E_3	E_2	E_1	E_0	D	C	B	A
0	0	0	1	1	0	0	0	0
1	0	1	0	0	0	0	0	1
2	0	1	0	1	0	0	1	0
3	0	1	1	0	0	0	1	1
4	0	1	1	1	0	1	0	0
5	1	0	0	0	0	1	0	1
6	1	0	0	1	0	1	1	0
7	1	0	1	0	0	1	1	1
8	1	0	1	1	1	0	0	0
9	1	1	0	0	1	0	0	1

Fig. 6.41. Conversion table: excess-3-to-BCD.

E_3	E_2	E_1	E_0
0	0	0	0
0	0	0	1
0	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1

Figure 6.42

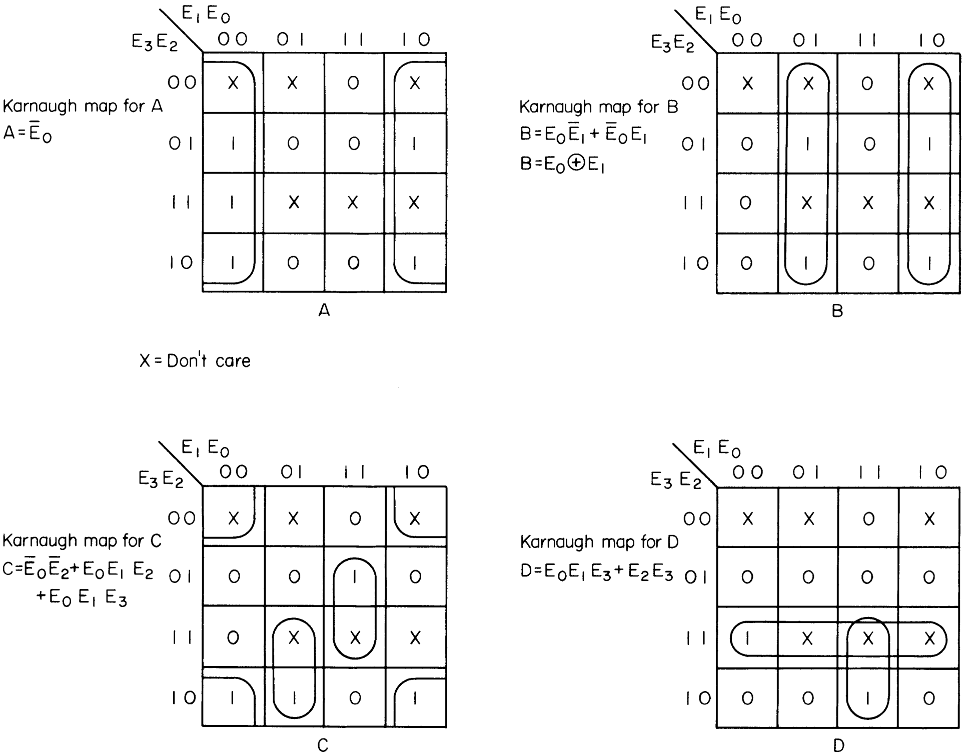

Karnaugh map for A
$A=\bar{E}_0$
A
Karnaugh map for B
$B=E_0\bar{E}_1+\bar{E}_0E_1$
$B=E_0\oplus E_1$
B
X = Don't care
Karnaugh map for C
$C=\bar{E}_0\bar{E}_2+E_0E_1E_2+E_0E_1E_3$
C
Karnaugh map for D
$D=E_0E_1E_3+E_2E_3$
D
E_1E_0
E_3E_2
00
01
11
10

Figure 6.43

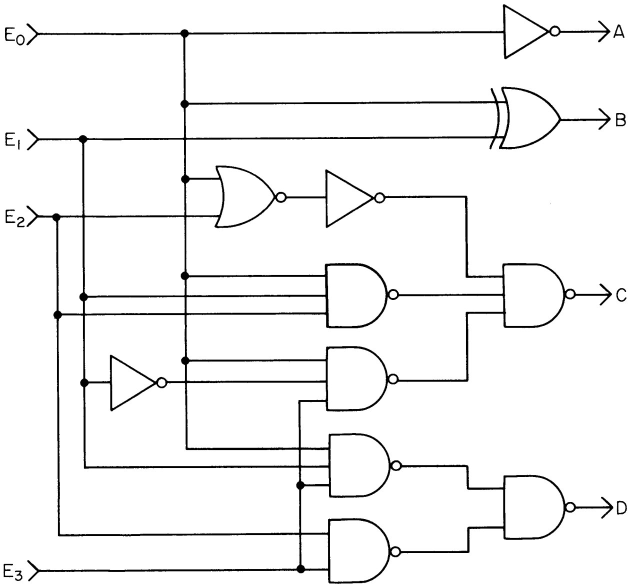


Fig. 6.44. Excess-3-to-BCD converter.

Decimal-to-BCD Converter. Decimal signals of nonelectronic systems are often available as 10 output lines. If a logical 1 is the active state of the line (see Fig. 6.48), the circuit of Fig. 6.49 may be used for the conversion to BCD. If the active state of a line is a logical 0 (see Fig. 6.50), the circuit of Fig. 6.51 may be used for the conversion.

Binary-to-BCD and BCD-to-binary Conversion. The converters described are of the *asynchronous* (nonclocked) type and consist solely of combinational logic gate arrays. The conversion technique used is based upon the methods described by J. F. Couleur and others.†

BINARY-TO-BCD CONVERTER. A binary number N_B is usually expressed in the form

$$N_B = b_{(n-1)}, b_{(n-2)}, \ldots, b_1, b_0$$

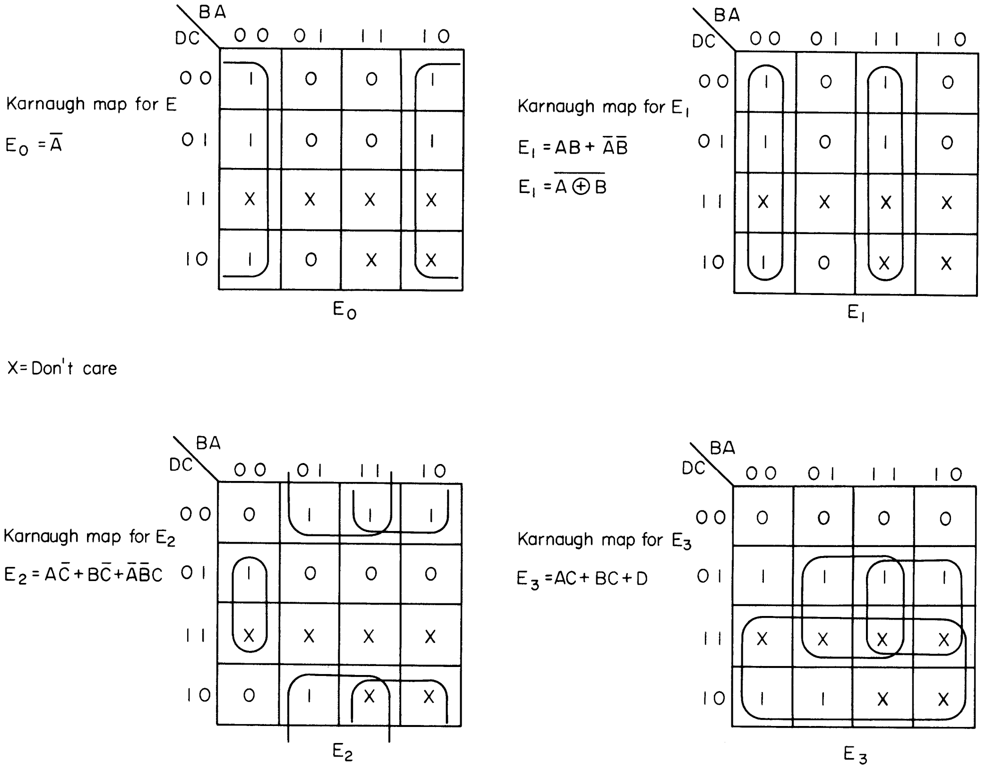
where n is the number of bits, and the b's are either 1s or 0s. The decimal

---

†See references 5, 6, and 7 of Bibliography at end of chapter.

D	C	B	A
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Figure 6.45



Karnaugh map for E

$E_0 = \bar{A}$

DC \ BA	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

E_0

Karnaugh map for E_1

$E_1 = AB + \bar{A}\bar{B}$

$E_1 = \overline{A \oplus B}$

DC \ BA	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

E_1

X= Don't care

Karnaugh map for E_2

$E_2 = A\bar{C} + B\bar{C} + \bar{A}\bar{B}C$

DC \ BA	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

E_2

Karnaugh map for E_3

$E_3 = AC + BC + D$

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

E_3

Figure 6.46

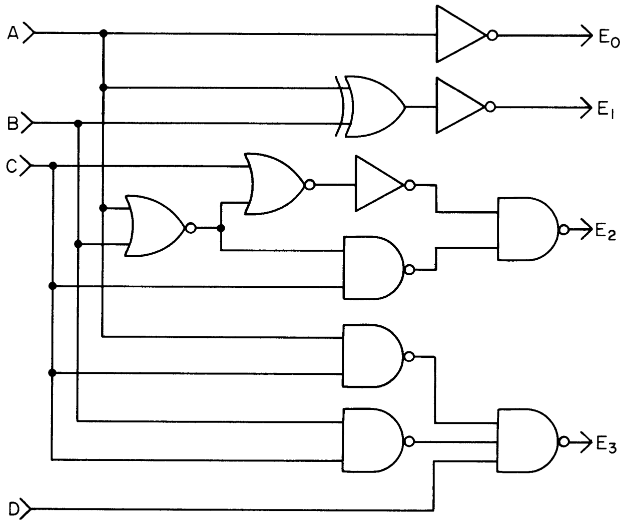


Fig. 6.47. BCD-to-excess-3 converter.

Conversion table: 10 lines to BCD, active 1s

Line 9	Line 8	Line 7	Line 6	Line 5	Line 4	Line 3	Line 2	Line 1	Line 0		D	C	B	A
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	2	0	0	1	0
0	0	0	0	0	0	1	0	0	0	3	0	0	1	1
0	0	0	0	0	1	0	0	0	0	4	0	1	0	0
0	0	0	0	1	0	0	0	0	0	5	0	1	0	1
0	0	0	1	0	0	0	0	0	0	6	0	1	1	0
0	0	1	0	0	0	0	0	0	0	7	0	1	1	1
0	1	0	0	0	0	0	0	0	0	8	1	0	0	0
1	0	0	0	0	0	0	0	0	0	9	1	0	0	1

Fig. 6.48. Conversion table: 10-lines to BCD, active 1s.

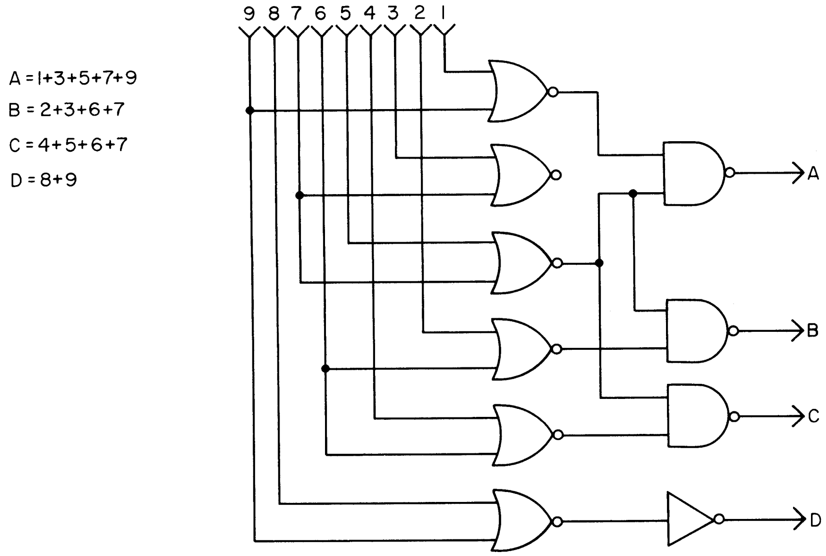


Fig. 6.49. 10-line decimal-to-BCD converter, for active 1s on input lines.

Conversion table: 10 lines to BCD, active 0s

Line 9	Line 8	Line 7	Line 6	Line 5	Line 4	Line 3	Line 2	Line 1	Line 0		D	C	B	A
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	1	1	0	0	0	1
1	1	1	1	1	1	1	0	1	1	2	0	0	1	0
1	1	1	1	1	1	0	1	1	1	3	0	0	1	1
1	1	1	1	1	0	1	1	1	1	4	0	1	0	0
1	1	1	1	0	1	1	1	1	1	5	0	1	0	1
1	1	1	0	1	1	1	1	1	1	6	0	1	1	0
1	1	0	1	1	1	1	1	1	1	7	0	1	1	1
1	0	1	1	1	1	1	1	1	1	8	1	0	0	0
0	1	1	1	1	1	1	1	1	1	9	1	0	0	1

Fig. 6.50. Conversion table: 10-lines to BCD, active 0s.

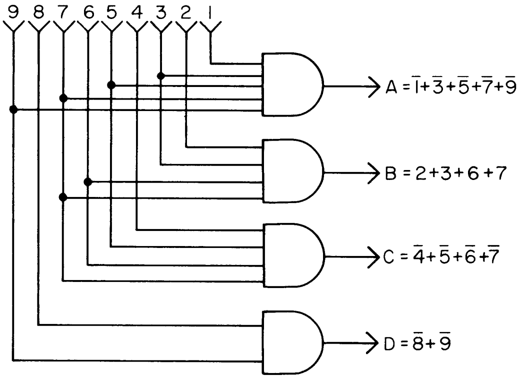

9 8 7 6 5 4 3 2 1
A = $\bar{1}+\bar{3}+\bar{5}+\bar{7}+\bar{9}$
B = $\bar{2}+\bar{3}+\bar{6}+\bar{7}$
C = $\bar{4}+\bar{5}+\bar{6}+\bar{7}$
D = $\bar{8}+\bar{9}$

Fig. 6.51. 10-line decimal-to-BCD converter, for active 0s on input lines.

value N_D of N_B is

$$N_D = b_{(n-1)} \cdot 2^{(n-1)} + b_{(n-2)} \cdot 2^{(n-2)} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0$$

The expression for N_D can be written in a form known as *nested multiplication by 2*. For example, the decimal value N_D of a 5-bit binary number is normally written as

$$N_D = b_4 \cdot s^4 + b_3 \cdot 2^3 + b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0$$

As a nested multiplication by 2, the expression for N_D will be

$$N_D = \{[(b_4 \cdot 2 + b_3) \cdot 2 + b_2] \cdot 2 + b_1\} \cdot 2 + b_0$$

The nested multiplication by 2 can be carried out by shifting the binary number N_B, most significant bit (MSB) first $b_{(n-1)}$, into a register at its least significant end. The nested multiplication is completed when the least significant bit (LSB) b_0 of N_B is shifted into the least significant stage of the register (see Fig. 6.52).

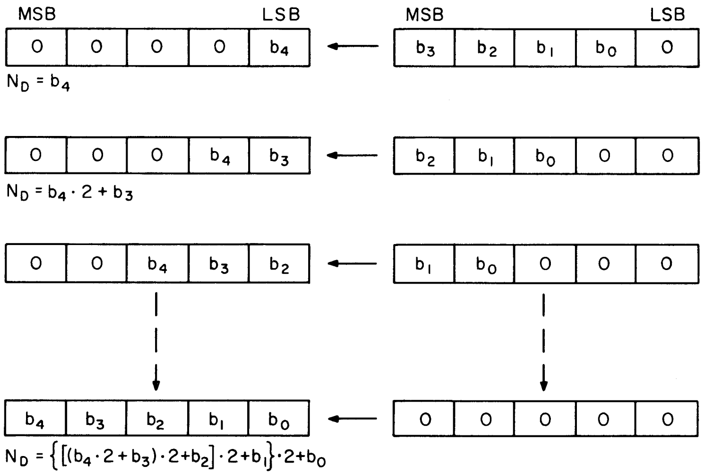

MSB
LSB
MSB
LSB
0 0 0 0 b_4 ← b_3 b_2 b_1 b_0 0
$N_D = b_4$
0 0 0 b_4 b_3 ← b_2 b_1 b_0 0 0
$N_D = b_4 \cdot 2 + b_3$
0 0 b_4 b_3 b_2 ← b_1 b_0 0 0 0
b_4 b_3 b_2 b_1 b_0 ← 0 0 0 0 0
$N_D = \{[(b_4 \cdot 2 + b_3) \cdot 2 + b_2] \cdot 2 + b_1\} \cdot 2 + b_0$

Fig. 6.52. Nested multiplication by 2.

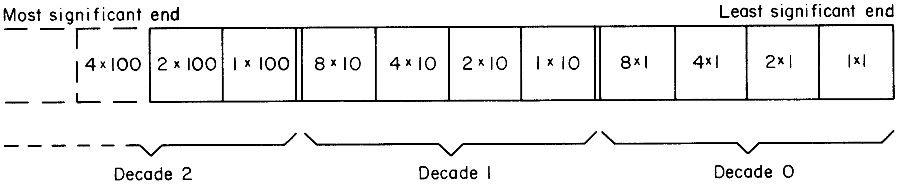


Fig. 6.53. BCD register value assignment.

Upon completion of the shifting in, N_B is stored in the register in *straight binary* form. To convert N_B into a BCD number, the value assignment of the register stages must be changed, and appropriate corrections must be made during the shifting. For this purpose, the register stages must be divided into groups of four consecutive stages, beginning with the least significant end of the register. Each group of four register stages represents a decade. Within a decade, starting at the least significant side of the decade, binary values are assigned to the four stages. The result is a value assignment as illustrated in Fig. 6.53, with the understanding that a decade *cannot* contain a number greater than 9. According to binary value assignment within a decade, each shift to the left in Fig. 6.53 represents a multiplication by 2. However, if a logical 1 passes the boundary between two decades—i.e., if a logical 1 is shifted from the 8-assigned stage of any decade to the 1-assigned stage of the next higher decade—its value increases only from 8 to 10 (instead of increasing from 8 to 16). Apparently, a value of 6 is lost. This loss must be corrected *after* the shift has taken place by adding 6 to the value of those decades from which a logical 1 has been shifted to a higher decade. This correction can also be made *before* the shift. In that case, whenever a logical 1 is present in the 8-assigned stage of a decade *before* the shift, 3 must be added to the value of that decade *prior* to the shift. This will result in a value increase by 6 after the shift. Another correction must be made if a decade, *after* shifting, contains a number greater than 9. In such a case, a value of 10 must be subtracted from that decade, and a value of 1 added to the next higher decade. Correction for this situation can also be made *before* the shift has taken place. In that event if a decade contains a number (greater than 4) that after shifting will produce a number greater than 9 *prior* to shifting, 5 must be subtracted from the number in that decade, and provision must be made for adding a value of 1 to the number of the next higher decade *after* the shift. This provision can be made by entering a logical 1 into the 8-stage of the decade with a number greater than 4 *prior* to shifting. Consequently, the correction for a decade which contains a number greater than 4 *before* the shift consists of subtracting 5 and adding 8, resulting in the addition of 3.

Thus, it turns out that all necessary corrections will take place automatically if, *prior* to shifting, 3 is added to the values of those decades which contain a number greater than 4. The described conversion procedure is shown in Fig. 6.54 for a 6-bit binary number.

The binary-to-BCD conversion as described is often implemented with shift registers and adders. This serial synchronous approach makes a parallel-to-serial input conversion necessary. In many cases a nonclocked approach, consisting of

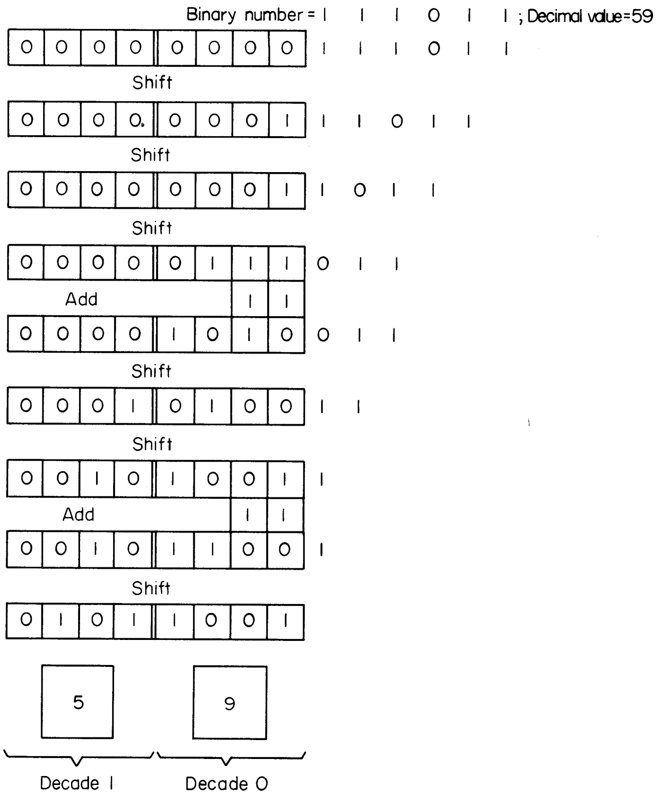

Binary number = 1 1 1 0 1 1 ; Decimal value=59
0 0 0 0 0 0 0 0 1 1 1 0 1 1
Shift
0 0 0 0 0 0 0 1 1 1 0 1 1
Shift
0 0 0 0 0 0 0 1 1 0 1 1
Shift
0 0 0 0 0 1 1 1 0 1 1
Add 1 1
0 0 0 0 1 0 1 0 0 1 1
Shift
0 0 0 1 0 1 0 0 1 1
Shift
0 0 1 0 1 0 0 1 1
Add 1 1
0 0 1 0 1 1 0 0 1
Shift
0 1 0 1 1 0 0 1
5
9
Decade 1
Decade 0

Fig. 6.54. Example of binary-to-BCD conversion algorithm.

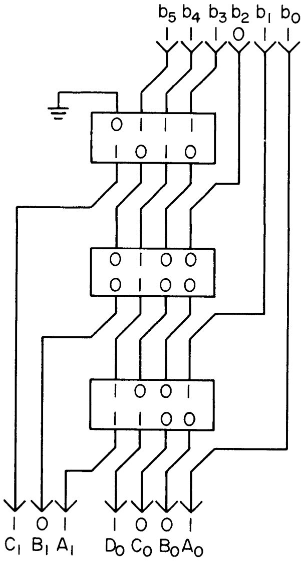

b_5 b_4 b_3 b_2 b_1 b_0
1 1 1 0 1 1
0 1 1 1
1 0 1 0
0 1 0 0
0 1 0 0
1 0 0 1
1 1 0 0
1 0 1
C_1 B_1 A_1
1 0 0 1
D_0 C_0 B_0 A_0

Fig. 6.55. Nonclocked configuration for binary-to-BCD conversion.

identical logic gate arrays, offers such advantages as parallel inputs that eliminate parallel-to-serial conversion, higher conversion speeds, and simple expansion for a greater number of bits.

In nonclocked systems, shifting can be achieved with so-called *hard-wiring between separate levels.* For each shift operation, a separate level or logic gate array is used. A nonclocked configuration for the conversion example of Fig. 6.54 is shown in Fig. 6.55. The first determination as to whether the shifted-in binary number is greater than 4 takes place *after* 3 bits have been shifted in. Because of the parallel input capability, these first three shifts can be eliminated in a nonclocked configuration. Adders are not necessary if the logic gate arrays are implemented in such a way that for numbers greater than 4 at the input the corrected numbers (input numbers plus 3) appear at the output. It will be obvious that the shifted-in number to be corrected can never be greater than 9. Thus, the identical logic gate arrays for a nonclocked implementation follow the conversion table of Fig. 6.56. The input states equivalent to the numbers 10 to 15 inclusive are "don't care" states, and the corresponding cells in the Karnaugh maps (Fig. 6.57) may be made 1s or 0s at free choice.

Thus, the logic expressions for the basic cell of a nonclocked binary-to-BCD converter are

$$K = E\bar{G}\bar{H} + \bar{E}FG + \bar{E}H$$

$$L = EF + \bar{E}H + F\bar{G}$$

$$M = EH + \bar{E}\bar{F}G$$

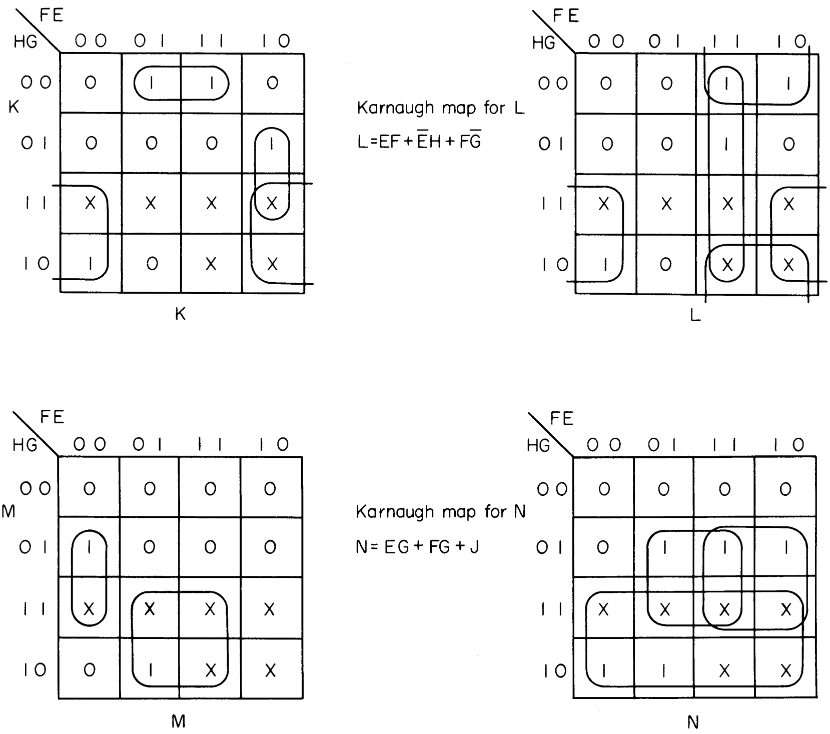
$$N = EG + FG + H$$

An implementation of the basic cell is shown in Fig. 6.58.

The basic cells must be cascaded to convert binary numbers with more than 4 bits. How the basic cells must be connected can be derived from the binary-to-BCD conversion diagram of Fig. 6.59. Each 4-bit rectangle with a plus sign in it represents

	Input				Output			
	H	G	F	E	N	M	L	K
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0
10	1	0	1	0	Don't care states			
11	1	0	1	1				
12	1	1	0	0				
13	1	1	0	1				
14	1	1	1	0				
15	1	1	1	1				

Figure 6.56



Karnaugh map for K

$K = E\bar{G}\bar{H} + \bar{E}FG + \bar{E}H$

HG \ FE	00	01	11	10
00	0	1	1	0
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

K

Karnaugh map for L

$L = EF + \bar{E}H + F\bar{G}$

HG \ FE	00	01	11	10
00	0	0	1	1
01	0	0	1	0
11	X	X	X	X
10	1	0	X	X

L

X = Don't care

Karnaugh map for M

$M = EH + \bar{E}\bar{F}G$

HG \ FE	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

M

Karnaugh map for N

$N = EG + FG + J$

HG \ FE	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

N

Figure 6.57

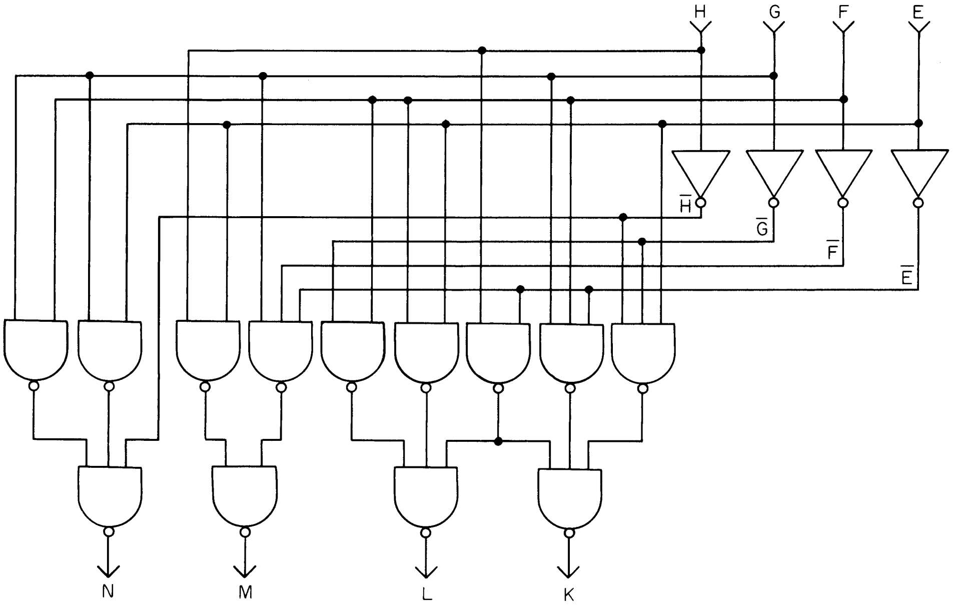


Fig. 6.58. Basic cell for a nonclocked binary-to-BCD converter.

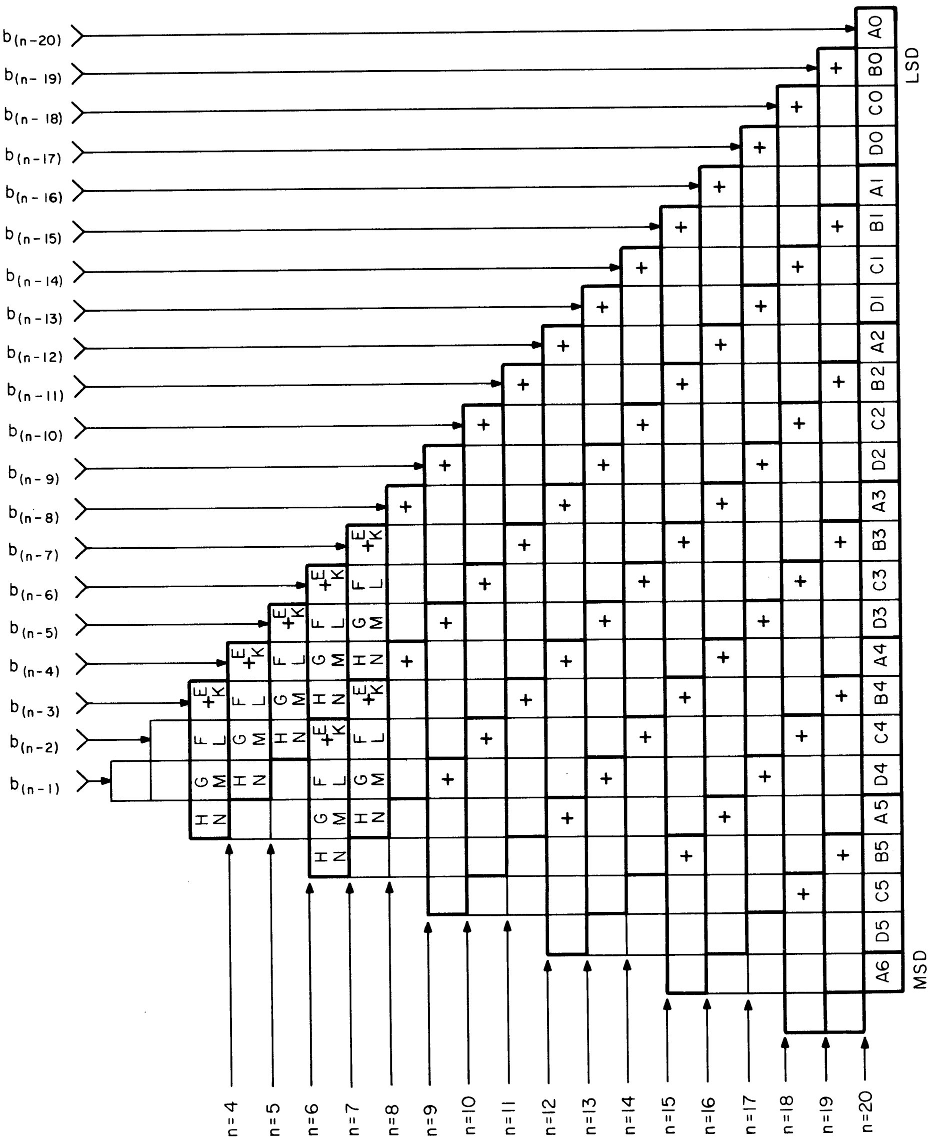

b(n-20)
b(n-19)
b(n-18)
b(n-17)
b(n-16)
b(n-15)
b(n-14)
b(n-13)
b(n-12)
b(n-11)
b(n-10)
b(n-9)
b(n-8)
b(n-7)
b(n-6)
b(n-5)
b(n-4)
b(n-3)
b(n-2)
b(n-1)
E
K
F
L
G
M
H
N
+
A0
B0
C0
D0
A1
B1
C1
D1
A2
B2
C2
D2
A3
B3
C3
D3
A4
B4
C4
D4
A5
B5
C5
D5
A6
LSD
MSD
n=4
n=5
n=6
n=7
n=8
n=9
n=10
n=11
n=12
n=13
n=14
n=15
n=16
n=17
n=18
n=19
n=20

Fig. 6.59. Binary-to-BCD conversion diagram.

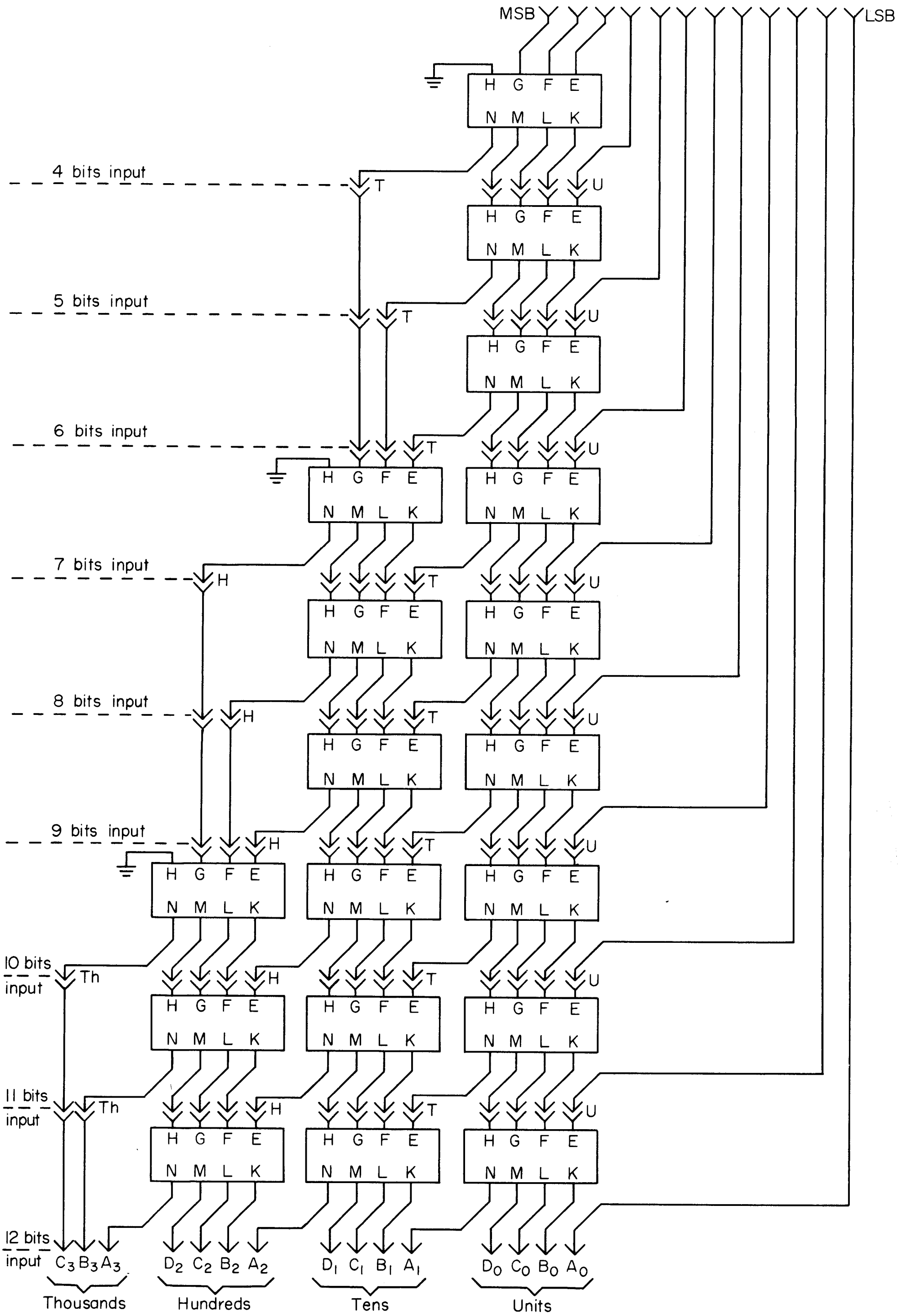

MSB
LSB
H G F E
N M L K
4 bits input
5 bits input
6 bits input
7 bits input
8 bits input
9 bits input
10 bits input
11 bits input
12 bits input
T
U
H
Th
C_3 B_3 A_3
D_2 C_2 B_2 A_2
D_1 C_1 B_1 A_1
D_0 C_0 B_0 A_0
Thousands
Hundreds
Tens
Units

Fig. 6.60. *n*-bit nonclocked binary-to-BCD converter.

a conversion step as well as a basic cell. The input and output assignments of the basic cells are shown for several conversion steps; they are identical throughout the whole diagram. Unused inputs must be connected to a logical 0 (ground). A practical implementation with basic cells for conversions of up to 12 bits is shown in Fig. 6.60.

BCD-to-binary Converter. For BCD-to-binary conversion, the BCD number is shifted out of the least significant end of the BCD register (see Fig. 6.53). Shifting in this direction represents a division by 2 with every shift. However, if a logical 1 passes the boundary between two decades—i.e., if a logical 1 is shifted from the 1-assigned stage of any decade to the 8-assigned stage of the next lower decade—its value reduces from 10 to 8 instead of from 10 to 5. Consequently, the resulting number is a value 3 higher than it should be. In such situations, a correction must be made *after* the shift by subtracting 3 from the value of those decades in which a logical 1 is shifted from higher decades. As a result, 3 must be subtracted from the values of those decades which *after* the shift contain a number equal to or greater than 8.

Notice that the correction is made *once* only: *after* the shift, because the new number obtained after correction (which is to be considered as the number *before* the next shift) may also be equal to or greater than 8. The new number, even though it may be equal to or greater than 8, is correct and must *not* be corrected again. The BCD-to-binary conversion procedure as described is shown in Fig. 6.61.

Like its binary-to-BCD counterpart, the BCD-to-binary converter is often imple-

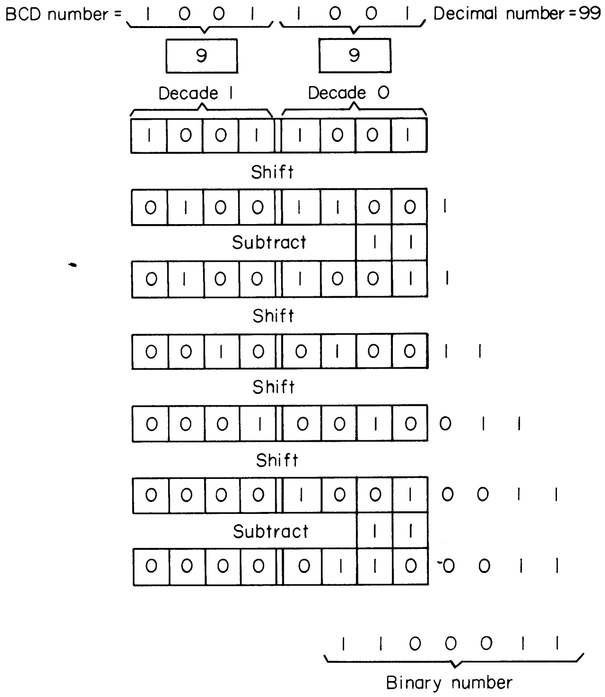

BCD number = 1 0 0 1 1 0 0 1 Decimal number = 99
9
9
Decade 1
Decade 0
1 0 0 1 1 0 0 1
Shift
0 1 0 0 1 1 0 0 1
Subtract 1 1
0 1 0 0 1 0 0 1 1
Shift
0 0 1 0 0 1 0 0 1 1
Shift
0 0 0 1 0 0 1 0 0 1 1
Shift
0 0 0 0 1 0 0 1 0 0 1 1
Subtract 1 1
0 0 0 0 0 1 1 0 0 0 1 1
1 1 0 0 0 1 1
Binary number

Fig. 6.61. Example of BCD-to-binary conversion.

mented with shift registers and adders in a serial synchronous manner. The BCD-to-binary converter described here, however, is of the nonclocked type with parallel input consisting of identical logic gate arrays. Again, shifting is achieved with hard-wiring between separate levels. The implementation of the identical logic gate arrays is such that for numbers equal to or greater than 8 at the input, the corrected numbers (input numbers minus 3) appear at the output. The 4-bit input number consists of the three most significant bits of a BCD decade and the least significant bit of the next higher BCD decade. Examination of the conversion procedure reveals that 5, 6, 7, 13, 14, and 15 never occur as input numbers so that the conversion follows Fig. 6.62. Consequently, the input states equivalent to the numbers 5 through 7 and 13 through 15 are "don't care" states, and thus the corresponding cells in the Karnaugh maps (Fig. 6.63) may be made 1s or 0s at free choice.

Thus, the basic cell of a nonclocked BCD-to-binary converter is determined by the logic expressions:

$$T = P \oplus S$$

$$U = \bar{P}Q + Q\bar{S} + P\bar{Q}S$$

$$V = R\bar{S} + P\bar{Q}S + \bar{P}\bar{R}S$$

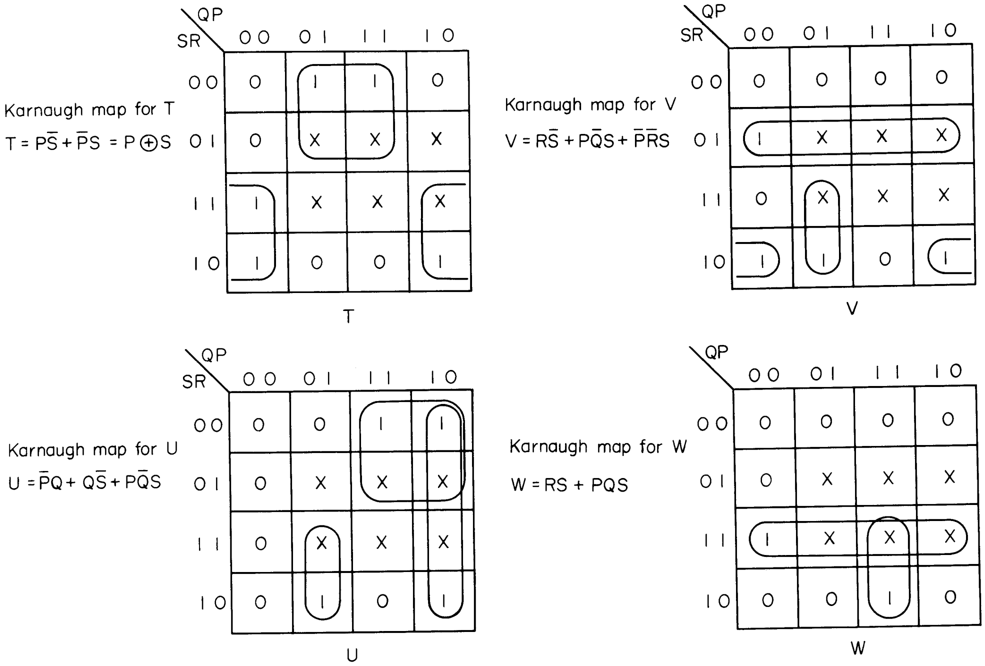
$$W = RS + PQS$$

An implementation of this basic cell is shown in Fig. 6.64.

The basic cells must be cascaded to convert BCD numbers into binary. The deviation for the correction of the basic cells can be obtained from the BCD-to-binary conversion diagram of Fig. 6.65. Each 4-bit rectangle that contains a plus sign represents a conversion step as well as a basic cell. The input and output assignments of the basic cells are shown for several conversion steps and are identical throughout the whole diagram. Unused inputs must be connected to a logical 0 (ground). A practical implementation for conversion of up to two BCD decades, using basic cells, is shown in Fig. 6.66.

	Input				Output			
	S	R	Q	P	W	V	U	T
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	0
3	0	0	1	1	0	0	1	1
4	0	1	0	0	0	1	0	0
5	0	1	0	1	Don't care states			
6	0	1	1	0				
7	0	1	1	1				
8	1	0	0	0	0	1	0	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	1	1
11	1	0	1	1	1	0	0	0
12	1	1	0	0	1	0	0	1
13	1	1	0	1	Don't care states			
14	1	1	1	0				
15	1	1	1	1				

Figure 6.62


QP
SR
00 01 11 10
T
V
U
W

Karnaugh map for T

$T = P\bar{S} + \bar{P}S = P \oplus S$

Karnaugh map for V

$V = R\bar{S} + P\bar{Q}S + \bar{P}\bar{R}S$

Karnaugh map for U

$U = \bar{P}Q + Q\bar{S} + P\bar{Q}S$

Karnaugh map for W

$W = RS + PQS$

Figure 6.63

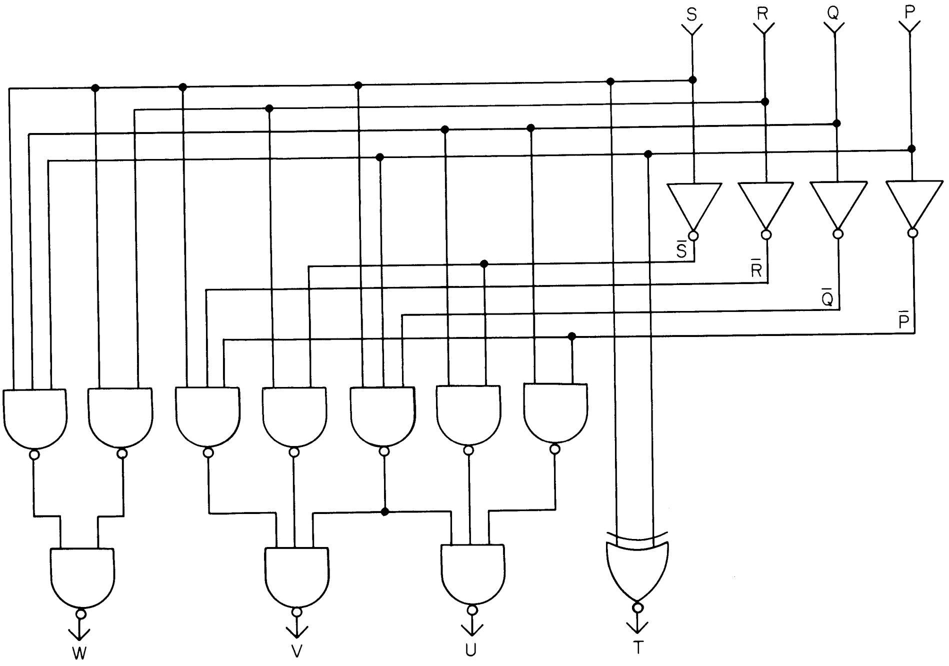

S
R
Q
P
$\bar{S}$
$\bar{R}$
$\bar{Q}$
$\bar{P}$
W
V
U
T

Fig. 6.64. Basic cell for a nonclocked BCD-to-binary converter.

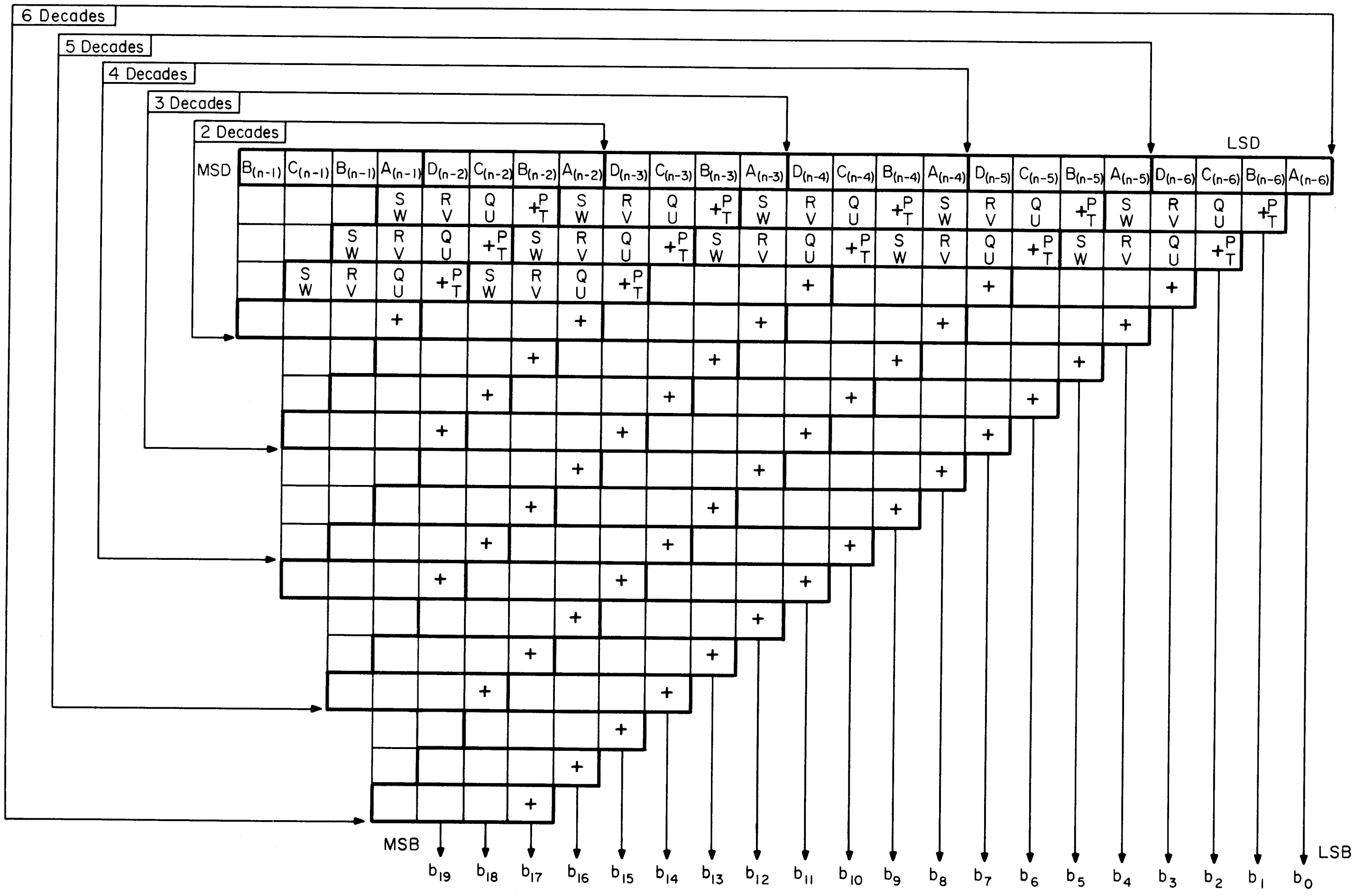

6 Decades
5 Decades
4 Decades
3 Decades
2 Decades
MSD
LSD
B(n-1) C(n-1) B(n-1) A(n-1) D(n-2) C(n-2) B(n-2) A(n-2) D(n-3) C(n-3) B(n-3) A(n-3) D(n-4) C(n-4) B(n-4) A(n-4) D(n-5) C(n-5) B(n-5) A(n-5) D(n-6) C(n-6) B(n-6) A(n-6)
P Q R S T U V W +
MSB
LSB
b19 b18 b17 b16 b15 b14 b13 b12 b11 b10 b9 b8 b7 b6 b5 b4 b3 b2 b1 b0

Fig. 6.65. BCD-to-binary conversion diagram.

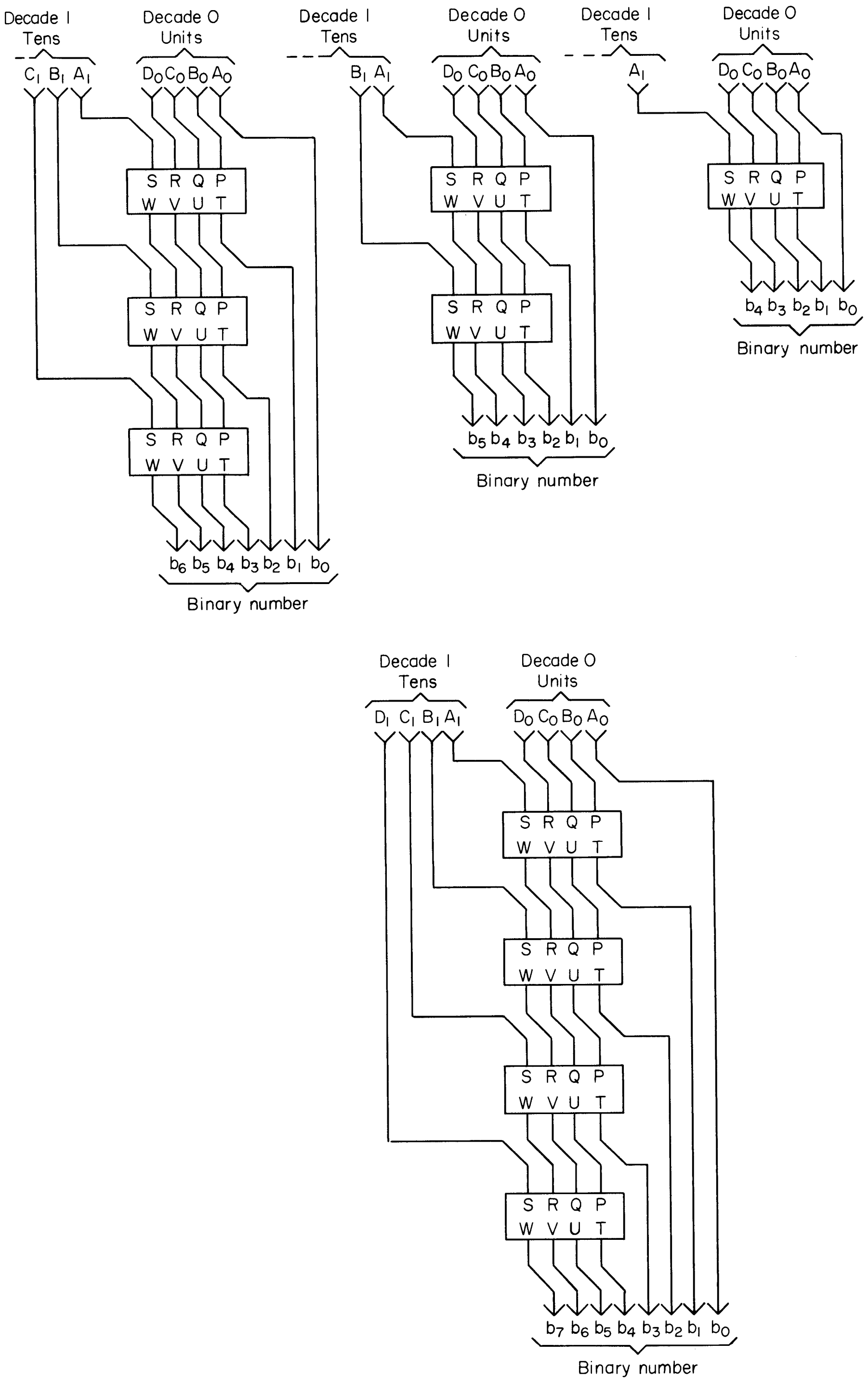

Decade 1
Tens
C_1 B_1 A_1
Decade 0
Units
D_0 C_0 B_0 A_0
S R Q P
W V U T
b_6 b_5 b_4 b_3 b_2 b_1 b_0
Binary number
Decade 1
Tens
B_1 A_1
Decade 0
Units
D_0 C_0 B_0 A_0
b_5 b_4 b_3 b_2 b_1 b_0
Binary number
Decade 1
Tens
A_1
Decade 0
Units
D_0 C_0 B_0 A_0
b_4 b_3 b_2 b_1 b_0
Binary number
Decade 1
Tens
D_1 C_1 B_1 A_1
Decade 0
Units
D_0 C_0 B_0 A_0
b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
Binary number

Fig. 6.66. Cascading basic cells to convert up to two BCD decades into binary.

Using four cascaded cells, two BCD decades can be converted into binary. It turns out that the configuration of four cascaded basic cells is repeatedly used when converting additional BCD decades. Consequently, the configuration of four cascaded basic cells, as shown in Fig. 6.67, may be considered as a standard building block. Cascading these building blocks for BCD-to-binary conversion is shown in Fig. 6.68 for up to five decades.

Binary Number Complementation. In digital arithmetic operations, a subtraction is performed by *adding* the *complement of the subtrahend* to the minuend. For this purpose, the *1's complement* as well as the *2's complement* may be used. The characteristic of the 1's complement is that, when added to the original number, an *all-1* number will result. Obviously, the 1's complement of a number is its *inverse.* The 2's complement is characterized by the fact that, when added to the original number, the result is an *all-0* number with a 1 overflowing from the most significant bit place. Using the 2's complement in subtraction has two advantages: no corrections with an end-around-carry are necessary, and representation of the number 0 is unambiguous, because it always appears as an all-0 number. In Fig. 6.70, the 1's and 2's complements of 4-bit numbers are shown. As previously mentioned, the 1's complement is the inverse of the original number. Thus, a 1's complementer consists solely of inverters. For a 2's complementer, the Karnaugh maps of Fig. 6.69 can be derived from Fig. 6.70.

A 4-bit 2's complementer is thus characterized by the logic expressions

$$T_0 = B_0$$

$$T_1 = B_0 \oplus B_1$$

$$T_2 = (B_0 + B_1) \oplus B_2$$

$$T_3 = (B_0 + B_1 + B_2) \oplus B_3$$

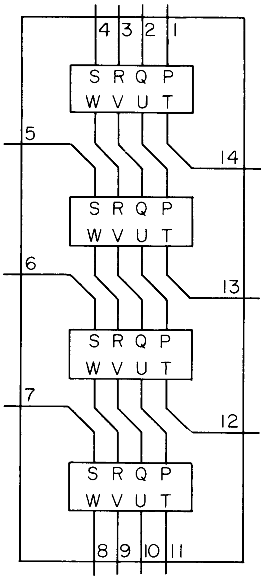


Fig. 6.67. Standard building block for BCD-to-binary converters.

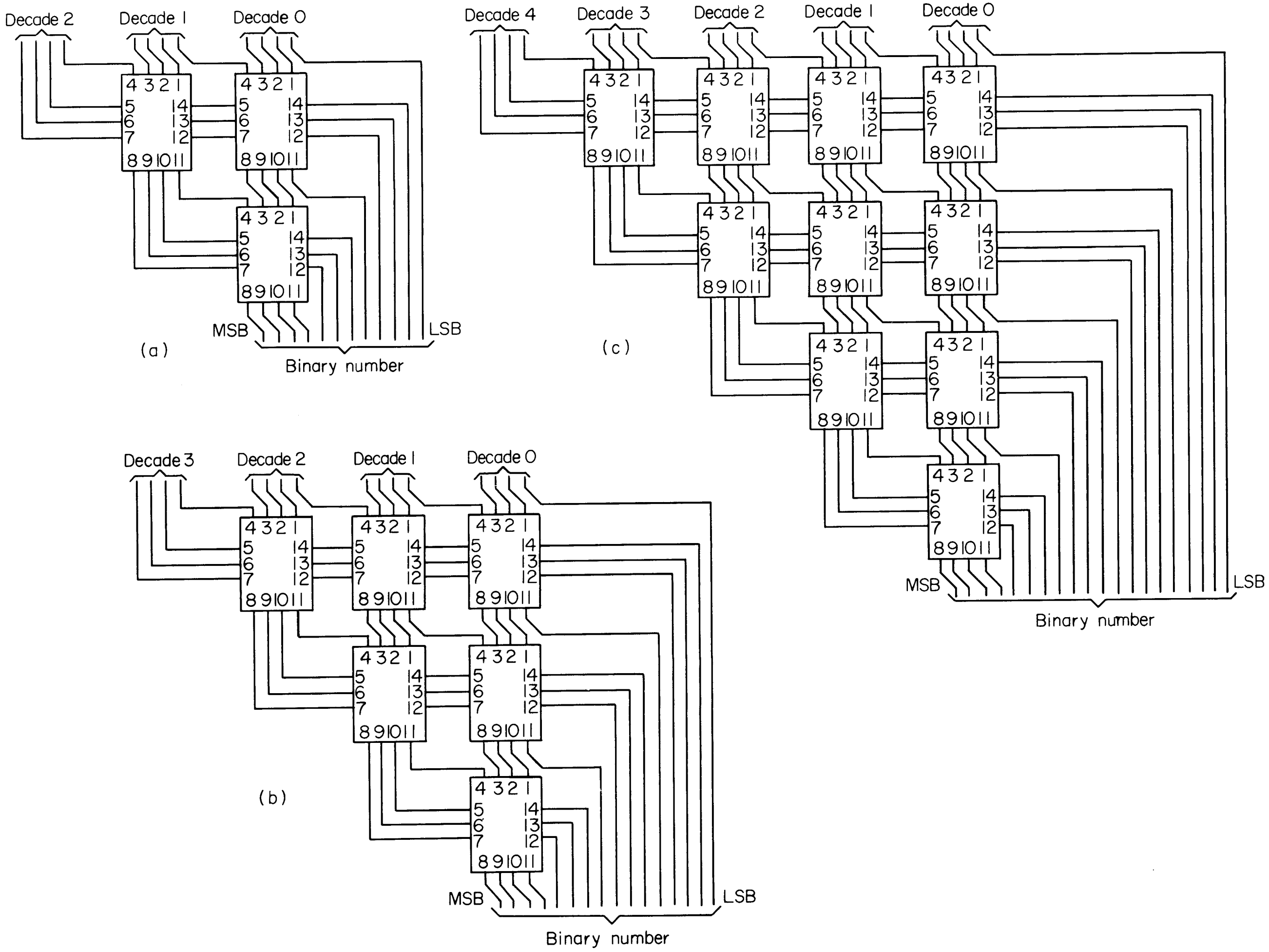
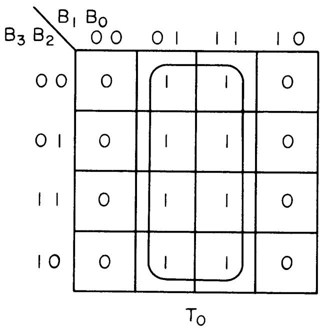

Decade 2
Decade 1
Decade 0
4 3 2 1
5 6 7
14 13 12
8 9 10 11
MSB
LSB
Binary number
(a)
Decade 4
Decade 3
Decade 2
Decade 1
Decade 0
(c)
Decade 3
Decade 2
Decade 1
Decade 0
(b)

Fig. 6.68. BCD-to-binary converters: *(a)* three decades; *(b)* four decades; *(c)* five decades.

Karnaugh map for T_0

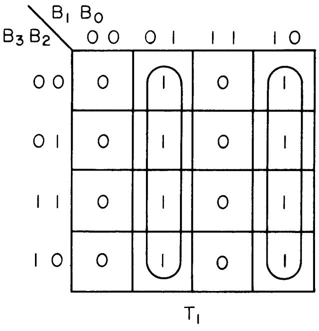
$T_0 = B_0$



$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

Karnaugh map for T_1

$T_1 = B_0\bar{B}_1 + \bar{B}_0B_1 = B_0 \oplus B_1$



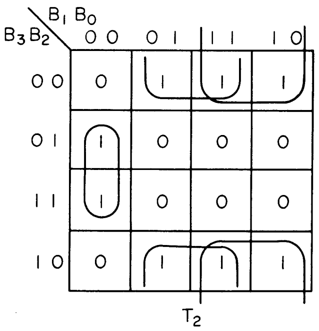
$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

Karnaugh map for T_2

$T_2 = B_0\bar{B}_2 + B_1\bar{B}_2 + \bar{B}_0\bar{B}_1B_2$

$T_2 = (B_0+B_1)\bar{B}_2 + \overline{(B_0+B_1)}B_2$

$T_2 = (B_0+B_1) \oplus B_2$



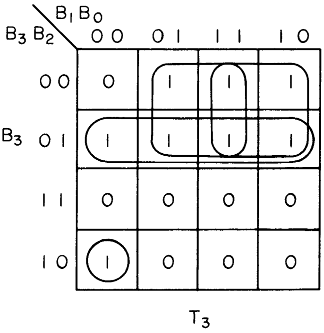
$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	1	0	0	0
10	0	1	1	1

Karnaugh map for T_3

$T_3 = B_0\bar{B}_3 + B_1\bar{B}_3 + B_2\bar{B}_3 + \bar{B}_0\bar{B}_1\bar{B}_2B_3$

$T_3 = (B_0+B_1+B_2)\bar{B}_3 + \overline{(B_0+B_1+B_2)}B_3$

$T_3 = (B_0+B_1+B_2) \oplus B_3$



$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	0	0	0

Figure 6.69

1's complement				Dec	Binary				2's complement			
C_3	C_2	C_1	C_0		B_3	B_2	B_1	B_0	T_3	T_2	T_1	T_0
1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	1	1	1	1	1
1	1	0	1	2	0	0	1	0	1	1	1	0
1	1	0	0	3	0	0	1	1	1	1	0	1
1	0	1	1	4	0	1	0	0	1	1	0	0
1	0	1	0	5	0	1	0	1	1	0	1	1
1	0	0	1	6	0	1	1	0	1	0	1	0
1	0	0	0	7	0	1	1	1	1	0	0	1
0	1	1	1	8	1	0	0	0	1	0	0	0
0	1	1	0	9	1	0	0	1	0	1	1	1
0	1	0	1	10	1	0	1	0	0	1	1	0
0	1	0	0	11	1	0	1	1	0	1	0	1
0	0	1	1	12	1	1	0	0	0	1	0	0
0	0	1	0	13	1	1	0	1	0	0	1	1
0	0	0	1	14	1	1	1	0	0	0	1	0
0	0	0	0	15	1	1	1	1	0	0	0	1

Figure 6.70

In general, a 2's complementer is characterized by the logic expression

$$T_n = (B_0 + B_1 + B_2 + \cdots + B_{(n-1)}) \oplus Bn$$

with the understanding that $T_0 = 0 \oplus B_0 = B_0$. An implementation of a 2's complementer is shown in Fig. 6.71.

It is obvious that the circuit of Fig. 6.71 can be used to convert a binary number to its 2's complement, as well as to convert a 2's-complemented number to its original

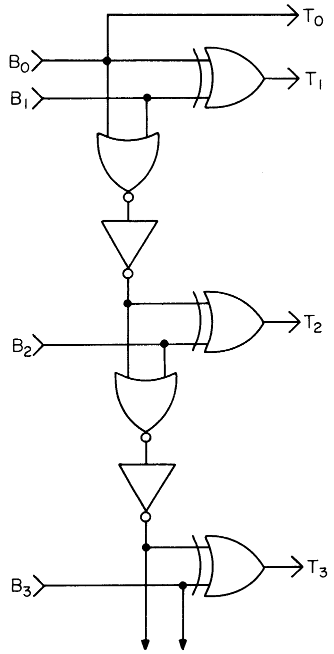


Fig. 6.71. 2's complementer.

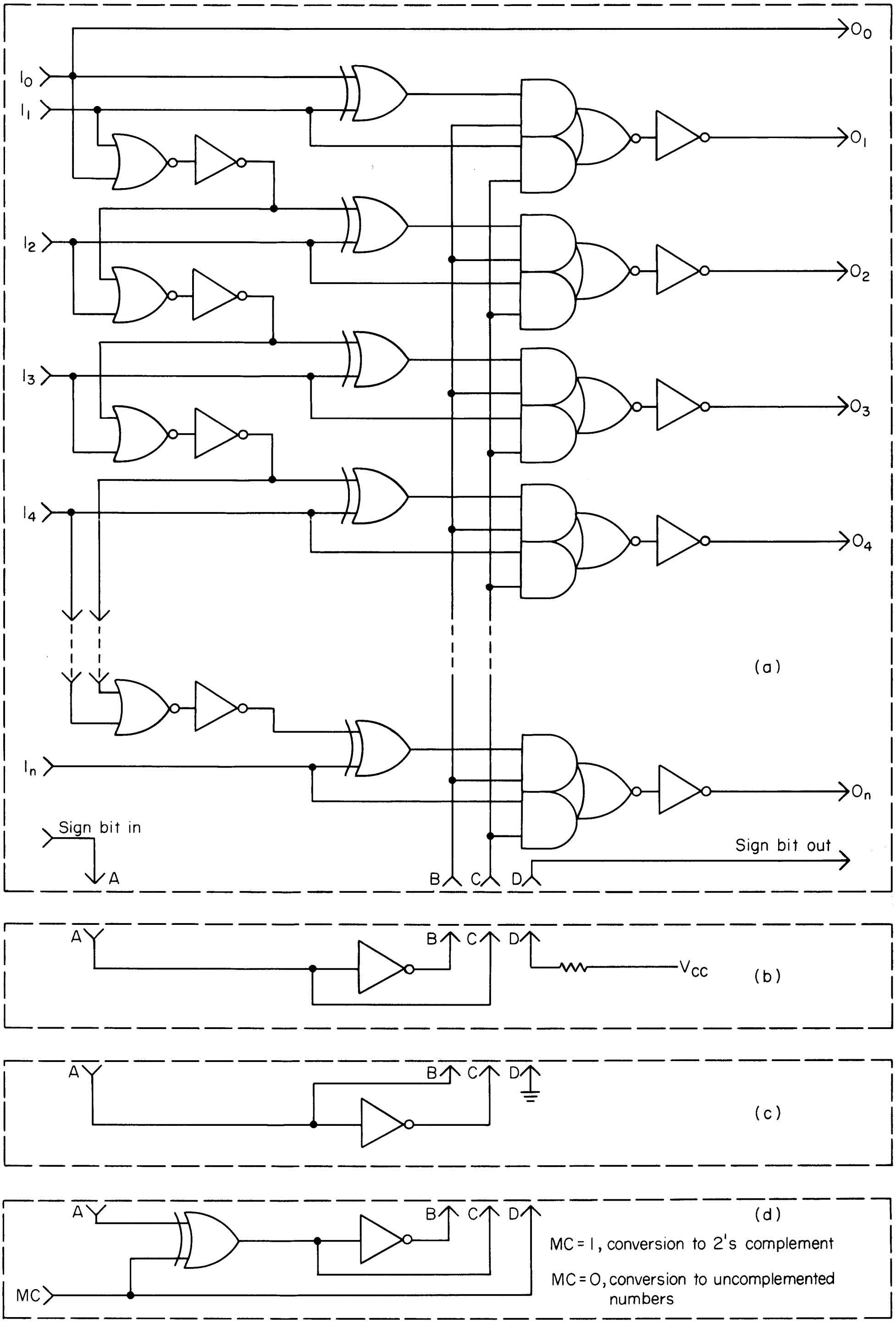

I0
I1
I2
I3
I4
In
O0
O1
O2
O3
O4
On
Sign bit in
A
B
C
D
Sign bit out
(a)
A
B
C
D
VCC
(b)
A
B
C
D
(c)
A
B
C
D
(d)
MC = 1, conversion to 2's complement
MC = 0, conversion to uncomplemented numbers
MC

Fig. 6.72. (*a*) and (*b*) conversion to 2's complement; (*a*) and (*c*) conversion to uncomplemented numbers; (*a*) and (*d*) controllable converter.

number. This is because the 2's complement of a 2's complement is the original number (see Fig. 6.70).

To distinguish uncomplemented (positive) numbers from complemented (negative) numbers, a *sign-bit* is added to the number on the *most significant* side. The sign-bit is a 0 for positive (uncomplemented) numbers and a 1 for negative (complemented) numbers.

For a subtraction, the subtrahend is entered into the arithmetic unit through a 2's complementer. There are situations, however, in which the subtrahend is available only in a 2's-complemented (negative) form which must *not* be complemented again. In those situations, the 2's complementer of Fig. 6.72*a* in conjunction with the control circuit of Fig. 6.72*b* can be used. This circuit will convert a number into its 2's complement only if its sign-bit is a 0. Numbers with sign-bits equal to 1 will be carried through without conversion. The sign-bit of the number at the output is always made a 1 to indicate negative (complemented) numbers. If the 2's complementer of Fig. 6.72*a* is used with the control circuit of Fig. 6.72*c*, it converts only negative (complemented) numbers with sign-bits equal to 1 to positive (uncomplemented) numbers. Positive numbers at the input will be carried through without conversion. The sign-bit of the number at the output is made a 0 to indicate positive (uncomplemented) numbers.

The circuits of Fig. 6.72*a* and *d* when used together perform both conversion modes under control of input MC. If $MC = 1$, they perform the same operation as the circuits of Fig. 6.72*a* and *b*; the operation of the circuits of Fig. 6.72*a* and *c* is performed if $MC = 0$.

BIBLIOGRAPHY

1. Humphrey, W. S., Jr.: "Switching Circuits: With Computer Applications," McGraw-Hill Book Company, New York, 1958.
2. Maley, G. A., and J. Earle: "The Logic Design of Transistor Digital Computers," Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.
3. McCluskey, E. J.: "Introduction to the Theory of Switching Circuits," McGraw-Hill Book Company, New York, 1965.
4. Wickes, W. E.: "Logic Design with Integrated Circuits," John Wiley & Sons, Inc., New York, 1968.
5. Couleur, J. F.: *IRE Transactions on Electronic Computers,* EC-7, No. 4, 1958, p. 313.
6. Sklar and MacDonald: EDN, April, 1967, p. 54.
7. Benedek and Moskowitz: *Electronic Design 21*, Oct. 10, 1968, pp. 58–64.

7

Flip-flops

A device that exhibits two different stable states is extremely useful as a memory element in a binary system. Any electrical circuit that has this characteristic falls into the category of devices commonly known as *flip-flops*. Some other names are *bistable multivibrator, multi, binary,* and *toggle*. All imply the 2-state characteristic of the device.

In its most basic form, a flip-flop consists of two cross-coupled inverting amplifiers. This simply means that the output of one amplifier is connected to the input of the other and vice versa. Figure 7.1*a* shows a circuit comprising two transistors and two resistors that demonstrates the cross-coupling idea. Obviously, if transistor $Q1$ is initially turned on (saturated) by applying a positive signal at its base, its collector will be at $V_{CE(\text{sat})}$ (typically 0.2 to 0.4 V). The collector of $Q1$ is then tied to the base of $Q2$ (Fig. 7.1*b*), turning $Q2$ off (cutoff). The collector of $Q2$ tries to rise to V_{CC}. This action only enhances the initial positive signal applied at the base of $Q1$, and we can then remove the initial signal and the circuit will maintain $Q1$ in the on state and $Q2$ in the off state indefinitely.

A positive signal injected at the base of $Q2$ turns it on. Repeating the sequence, $Q2$ turns on (saturates) and $Q1$ turns off, resulting in a second stable state. In

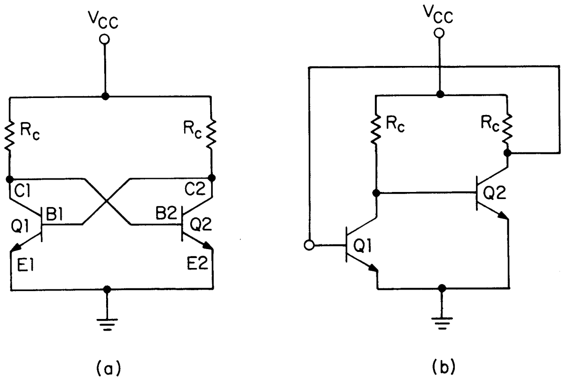


Fig. 7.1. Comparison of cross-coupled flip-flops: (*a*) conventional; (*b*) as cascaded amplifier.

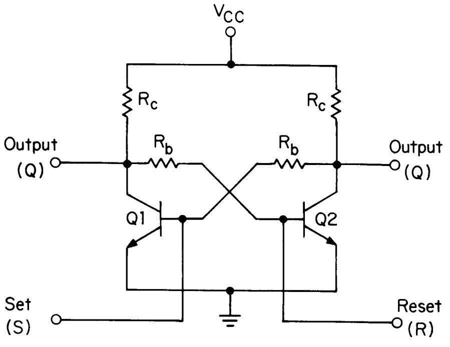


Fig. 7.2. Practical circuit for a discrete flip-flop.

general, the flip-flop just described can be called a 2-stage positive-feedback saturating amplifier.

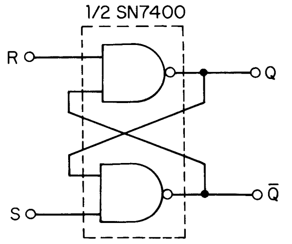
Although the circuit shown does have two stable states, it is evident that any signal output at either collector is very small. For example, when $Q1$ saturates, its base can only rise to a $V_{BE(\mathrm{sat})}$ of 1.0 V. If an output is taken at the collector of $Q2$, the voltage swing is approximately $1.0 - V_{CE(\mathrm{sat})}$ or $1.0 - 0.2$ to $0.4 \approx 0.6$ V. This is not really a usable signal level in most digital systems using saturated logic.

Figure 7.2 shows a more functional circuit which differs from the first mainly in the addition of two base resistors R_B and two control inputs. For values of $R_B = 10R_C$ the circuit will operate like the one in Fig. 7.1*a*. However, the output signals Q and $\overline{Q}$ (Q = noninverted output, $\overline{Q}$ = inverted output) have a voltage swing approximately equal to $V_{CC} - V_{CE(\mathrm{sat})}$. For $V_{CC} = 5.0$ V the output swing is ≈ 4.6 V. In actual practice this value is somewhat lower because of the resistor divider and load current being drawn, typically 3.8 to 4.2 V. It should be noted that for TTL flip-flops, the voltage swing for a V_{CC} of 5 V is typically 3.5 V.

When a positive voltage or pulse is applied to the *set* input, output Q is forced to the highest positive voltage or logical 1 state. When a positive voltage or pulse is applied to the *reset* input, output Q is forced to its lowest voltage or logical 0 state. The terms 1 and 0 are used to define the high and low output levels, respectively.

This circuit falls into a category of flip-flops known as latches (the circuit will be defined in greater detail later). It is important to note, however, that some form of latch is used in every flip-flop. The basic latch concept can be expanded to include such inputs as clocks, multiple data inputs, and variations in enabling or preset controls.

Flip-flops ranging from 1-bit storage elements to multibit arrays are useful in arithmetic sections of computers because of their high-speed capabilities. Flip-flops are used in all types of shift and buffer registers, including serial right-and-left-shift with parallel in and out capability, storage buffers for computer input-output systems, and alphanumeric indicator displays, and as accumulators in conjunction with adders to store and update data when clocked. Flip-flops are used also in all types of counter applications including up/down, presettable, gated, and any combination of these. Other uses include error detection and data conversion.



Truth table			
R	S	Q	$\bar{Q}$
0	0	1*	1*
0	1	1	0
1	0	0	1
1	1	No change	

*Not allowed, see text

Fig. 7.3. *R-S* NAND latch.

7.1 FLIP-FLOP TYPES

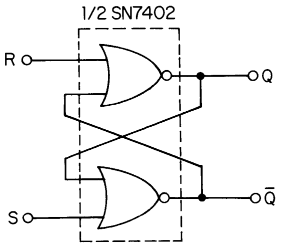
The four basic types of flip-flops are the *D, T, R-S,* and *J-K*. The *R-S* latch will be considered separately because it has one characteristic which is different from the other four.

The Basic Flip-flop. Historically, the *R-S* latch was probably the first type of flip-flop built and used for data storage because it is considerably simpler than any of the others. The circuit in Fig. 7.2 is an *R-S* latch using discrete components. If the transistors and resistors of Fig. 7.2 are replaced with two dual-input NAND gates (one-half SN7400), the circuit of Fig. 7.3 results. It has two control inputs and two outputs. The *R* (reset) and *S* (set) (also called *clear* and *preset* respectively) are termed asynchronous controls since the output changes immediately when either control input changes. For the gates shown, the 0s and 1s in the truth table represent 0.2 V and 3.3 V typically for positive logic notation.

From the truth table it is evident that four definite states exist for the *R-S* latch. Since $\bar{Q}$ is defined as the inverse of Q, however, the $R = S = 0$, $Q = \bar{Q} = 1$ state is not allowed. It should be noted that when an *R-S* latch is built as shown in Fig. 7.3, the assignment of inputs and outputs is purely arbitrary. This is not generally the case in the other types of flip-flops discussed later.

Figure 7.4 shows an *R-S* latch implemented with NOR gates (one-half SN7402) in place of NAND gates. From the truth table it is seen that for $S = 0$, $R = 1$, and $S = 1$, $R = 0$, the outputs are identical to the NAND latch of Fig. 7.3. The differences are purely a result of the differences in the NAND and NOR gate functions. Here again, the $R = S = 1$, $Q = \bar{Q} = 0$ state is not allowed.

As stated earlier, the latch is limited to one mode of operation, asynchronous.



Truth table			
R	S	Q	$\bar{Q}$
0	0	No change	
0	1	1	0
1	0	0	1
1	1	0*	0*

*Not allowed, see text

Fig. 7.4. *R-S* NOR latch.

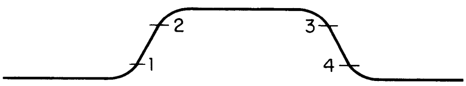
But it is often desirable to have some form of clock input to the flip-flop, so that the device may be operated simultaneously (synchronously) with all others in a system. It is this requirement which led to the development of the other types of flip-flops.

Synchronous Clocking. Since flip-flops described later have synchronous and clock inputs, a short discussion of these inputs is injected here. In general, the different types of synchronous (also referred to as "data") inputs determine the name of the flip-flop, while the different mechanisms for clocking can be used with any flip-flop type. A synchronous data input is one which does not cause an immediate change in the output. It requires the presence or occurrence of another input, the clock pulse, to generate the change of state. This action makes the device useful in a system application since one clock can be used to control a large number of flip-flops, causing them to change states synchronously and predictably. In TTL flip-flops there is only a single clock input, but the data inputs range from one to about ten, depending on the device and operation desired. The single clock input can be one of three general types: d-c or edge-triggered, a-c coupled, or master-slave.

The d-c or edge-triggered clock is one which causes flip-flop operation at a particular voltage when either a positive or a negative transition occurs. Either one or the other is recognized, but not both, for any chosen device. This type of clock enables the data inputs and transfers the data to the outputs simultaneously, resulting in a high-speed clocking technique that is relatively independent of clock rise and fall times. However, in TTL devices noise immunity decreases as the rise and fall times increase beyond 150 ns.

A-c coupled clocks are not used in any TI TTL device, but they are used extensively in DTL devices. An a-c coupled clock responds to a transition time rather than a voltage, since the clock is capacitively coupled internally to the latching mechanism in the flip-flop. These clocks can be either positive or negative in operation since they respond to one transition or the other. This technique is highly dependent on clock rise or fall times. Typically, values less than 200 ns are required for the device to operate. At values greater than this, operation is either intermittent or impossible.

The master-slave type is basically two latches connected serially. The first latch is called the master and the second is termed the slave. Normal action in clocking consists of four steps as shown in Fig. 7.5. Although the four points are shown as definite points on the waveform, the exact d-c points can be quite wide in tolerance and are not usually specified. The important feature of this type of clocking is that the data inputs are never directly connected to the outputs at any time during clocking; this provides total isolation of outputs from data inputs.



1. Isolate slave from master
2. Enable data inputs to master
3. Disable data inputs
4. Transfer data from master to slave

Fig. 7.5. Master-slave clock pulse.

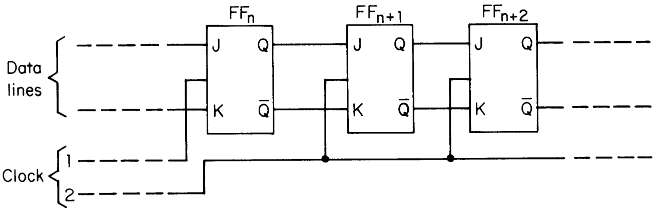


Fig. 7.6. Shift register (using *J-K* flip-flops).

The phenomenon called clock skew should be mentioned. Consider, for example, a shift register in which the outputs of one flip-flop are directly connected to the inputs of the following flip-flop, as shown in Fig. 7.6. The two clock lines are generated at the same point, but owing to loading conditions, two separate drivers are required. Figure 7.7 shows the clock generator and drivers. The waveforms suggest a possible condition that could exist if the drivers were not of equal delay. The term t_{PHL} is the propagation delay of the gates from the logical 1 to the logical 0 state, and Δt_{phl} represents the difference between the two gate delays plus capacitive delays due to wire routing and lengths. The shift register uses negative edge-triggered *J-K* flip-flops whose worst-case minimum propagation time is 10 ns. If the Δt_{phl} time in Fig. 7.7 is greater than 10 ns, the *n*th flip-flop in Fig. 7.6 could change states before the $(n + 1)$st flip-flop is ever clocked, resulting in possibly erroneous data being shifted into the $(n + 1)$st flip-flop. The maximum allowable clock skew in this system is 10 ns.

In general, the maximum allowable clock skew for a flip-flop can be expressed as

$$T_{skew(\max)} = T_{pd(FF)} - T_{h(FF)}$$

where $T_{pd(FF)}$ is the worst-case minimum propagation delay of the flip-flop, and $T_{h(FF)}$ is the worst-case maximum hold time of the flip-flop. The hold time is defined as the length of time the data inputs must be held stable after the clock pulse has dropped below a given point (usually the 50 percent point, or about 1.5-V level for TTL) to ensure proper operation of the flip-flop. For TTL flip-flops (with only a few exceptions) the hold time will be 0.

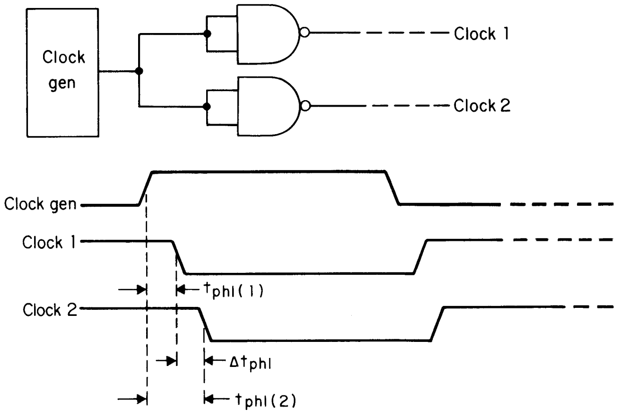


Fig. 7.7. Clock waveforms and generator.

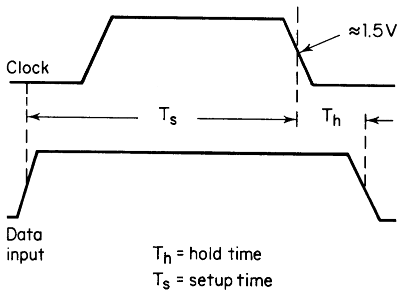


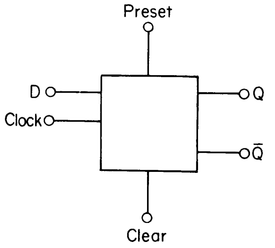
Fig. 7.8. Waveform illustrating setup and hold times.

Associated with hold time, setup time is defined as the length of time the data inputs must be stable before clocking the flip-flop. This time is measured from the 50 percent point on the clock pulse edge which activates the outputs. Figure 7.8 shows how setup and hold times are measured with respect to the clock pulse.

D-type Flip-flop. The *D*-type flip-flop is characterized by a single data (*D*) input and a clock input. It may have either or both Q and $\overline{Q}$ outputs available. It can have asynchronous inputs as shown in Fig. 7.9. Although both the preset and clear inputs are shown, they may or may not be present on some *D*-type flip-flops. Usually the preset and clear are overriding controls; that is, when either of these inputs is present (in this case logical 0), the operation of the device is inhibited and the output goes to either a 0 or 1, depending on which input is present. The operation of the flip-flop is shown by the truth table of Fig. 7.9; the state of the *D* input is shown prior to clocking, and the state of the *Q* output is shown just after clocking.

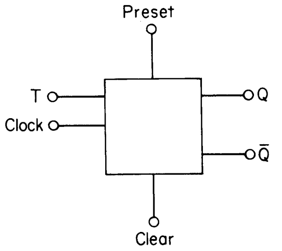
D-type flip-flops can use any one of the three types of clocking techniques mentioned earlier, but a special case of edge-triggering is sometimes used with this type of flip-flop, in which case it is called a *D*-type latch. The clocking in this case is still done on an edge, but .while the clock is high, the *E* input is directly coupled to the output *Q*. During this time any changes at the *D* input are immediately reflected at the *Q* output. When the clock falls, the *Q* output holds the state of the *D* input prior to the clock edge until the clock goes high again. The clock actually operates as an enable input to a latch. This kind of flip-flop is useful primarily in data storage and register applications, where temporary data storage is required.

T-type Flip-flop. The *T*-type flip-flop also has only a single data (*T*) input and a clock input. It may also have asynchronous controls, clear, preset or both, and one or both of Q and $\overline{Q}$ outputs, as shown in Fig. 7.10. Again, any one of



Truth table	
t_n	t_{n+1}
D	Q
0	0
1	1

Fig. 7.9. *D* flip-flop.



Truth table	
t_n	t_{n+1}
T	Q
0	Q_n
1	$\bar{Q}_n$

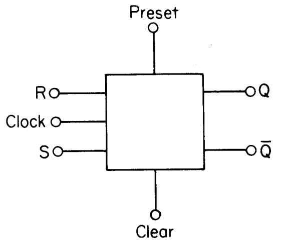
Fig. 7.10. *T* flip-flop.

the three clocking techniques can be used. The truth table shows that the operation of the flip-flop is quite simple. When the *T* input is in the 0 state prior to a clock pulse, the *Q* output will not change with clocking. When the *T* input is at a 1 level prior to clocking, the *Q* output will be in the $\bar{Q}_n$ state after clocking. In other words, if the *T* input is a logical 1 and the device is clocked, the output will change state regardless of what the output was prior to clocking. This is called *toggling;* hence the name *T* flip-flop.

The *T* flip-flop is not generally available as such but is usually generated from one of the other types, as is shown later. It is most often seen in counters and sequential counting networks because of its inherent divide-by-2 capability. When a clock pulse is applied, the output changes state once every input cycle, thus completing one cycle for every two input cycles. This is the action required in many cases for binary-coded counters.

R-S Flip-flop. The *R-S* or *S-R* type of flip-flop has either set or reset asynchronous input, or both, and a *Q* and $\bar{Q}$ output. Figure 7.11 shows an *R-S* flip-flop and its truth table. Notice that the first three states of this flip-flop are identical to the first three states of the *R-S* latch $\bar{Q}$ output in Fig. 7.4. The only difference is the clock input to the flip-flop. For $R = S = 0$, the *Q* output remains the same after clocking, a logical 1. For $R = 1$, $S = 0$, the *Q* output after clocking is a logical 0.

Note that when the *Q* output is in the logical 0 state and the *S* data input is at a 0 level, it does not matter whether the *R* input is a 0 or 1. The *Q* output at Q_{n+1} will always be a logical 0. Similarly, if Q_n is a logical 1 and the *R* input is a 0, regardless of the state of the *S* input the output Q_{n+1} is a logical 1. These conditions are significant when designing circuits since they reduce the amount of gating required in counting and decoding applications.

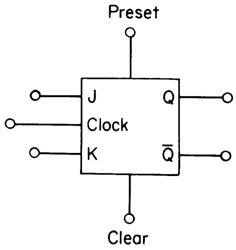


Truth table		
R	S	Q
0	0	Q_n
0	1	1
1	0	0
1	1	*

* Indeterminate

Clear = 0, Q = 0
Preset = 0, Q = 1

Fig. 7.11. *R-S* flip-flop.



Truth table		
t_n		t_{n+1}
J	K	Q
0	0	Qn
0	1	0
1	0	1
1	1	$\overline{Q}n$

Excitation table			
Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

X = Don't care

Fig. 7.12. *J-K* flip-flop.

The only factor that limits the use of the *R-S* flip-flop in actual applications is the indeterminate condition of the output when $R = S = 1$. Since the output is not predictable, it is necessary to avoid this input condition when clocking. When it cannot be avoided, a *J-K* flip-flop is used.

***J-K* Flip-flop.** The *J-K* flip-flop has two data inputs, *J* and *K*, and only a single clock input. Many *J-K* types have internal gates on the inputs to provide multiple *J* and *K* inputs, thereby eliminating the need for external gates in many applications. Figure 7.12 shows a *J-K* flip-flop with two asynchronous control inputs (preset and clear) and two outputs (Q and $\overline{Q}$). The truth table shows that all four possible states of the input generate defined output states. The truth table is identical to the *R-S* type ($J = S$, $K = R$) with one exception; the fourth condition is $J = K = 1$. When this input condition exists and the flip-flop is clocked, the output changes state. This is the same as the *T* type when the *T* input is at a logical 1. The flip-flop simply toggles. Figure 7.12 also presents an excitation table. This table is basically generated by an intuitive analysis of the truth table. If the state of the output Q_n prior to clocking is known and it is desired to have an output Q_{n+1} after clocking, the table shows which data input is necessary. The term shown in the table as X represents a "don't care" state. For example, when Q_n is a logical 0 and it is desired to have a 0 after clocking ($Q_{n+1} = 0$), it is necessary that $J = 0$ before clocking. The state of K does not matter. As noted for the *R-S* flip-flop, these "don't care" (X) states are valuable in logic design. They allow the designer to use fewer gates at the data inputs to control the gate operation while maintaining defined outputs for all input conditions. The *J-K* flip-flop is considered the most versatile available, and the variety of devices on the market reflects the trend toward its use.

Figure 7.13 shows a typical *J-K* master-slave flip-flop in a gate-type representation. This model makes it quite easy to see how the excitation table is generated. The Q output is directly gated with the K input, while the $\overline{Q}$ output is gated with the J input. The gates are NAND functions, and if any input is a logical 0, the output is a logical 1. Assume initially $Q = 0$ and $\overline{Q} = 1$. Output Q is gated with K and the clock at Z1. When the clock rises to a logical 1, Q holds the output of gate Z1 at a 1 level regardless of the state of the K input. A similar condition exists at Z4. However, when the clock rises to a logical 1 level (clear must be a 1 for flip-flop operation), $\overline{Q}$ is a logical 1. The flip-flop output at Q_{n+1} is determined only by the J input. The same analysis can be used for $Q = 1$ and $\overline{Q} = 0$. The state of the K input determines the output.

The J-K may have any of the three general types of clocking. The master-slave J-K has one rather unique characteristic: the setup time for the J and K inputs is always greater than or equal to the clock pulse width. Therefore, the J and K inputs must be held stable until the clock falls. If this condition is not fulfilled, the flip-flop operation may be erroneous.

Again using Fig. 7.13 and assuming $Q_n = 1$, $\overline{Q}_n = 0$, examine what happens if the K input changes from a logical 0 to logical 1 and back to a logical 0 while the clock is at logical 1 (the J input is inhibited since $\overline{Q} = 0$). When $K = 0$, the output of gate Z1 is 1 and does not change the state of the master latch, Z2 and Z3. If K momentarily rises to a 1 level, the output of Z1 goes to a logical 0, changing the state of the master. When K returns to a logical 0 level, Z1's output returns to logical 1; however, the master latch effectively "remembers" the 1 condition at the K input. When the clock falls to the 0 level, the q output changes states: $Q_{n+1} = 0$, $\overline{Q}_{n+1} = 1$.

When $Q_n = 0$ and $\overline{Q}_n = 1$, the J input controls the output. If the J input rises to a logical 1, while the clock is at logical 1, the master section will "remember" this state. After the clock returns to a 0 level, the outputs change to $Q_{n+1} = 1$ and $\overline{Q}_{n+1} = 0$. The excitation table can be used to give a general statement for J-K master-slave operation when the J and K inputs change with the clock high. If $Q_n = 0$ and J rises to a logical 1, Q_{n+1} will be 1. If $Q_n = 1$ and K rises to a logical 1, Q_{n+1} will be 0. It is this characteristic that causes the setup time for J-K master-slave flip-flops to be greater than or equal to the clock pulse width.

In recent years the J-K has increased in popularity owing to its wide flexibility and the decreasing price differential between it and other types of flip-flops.

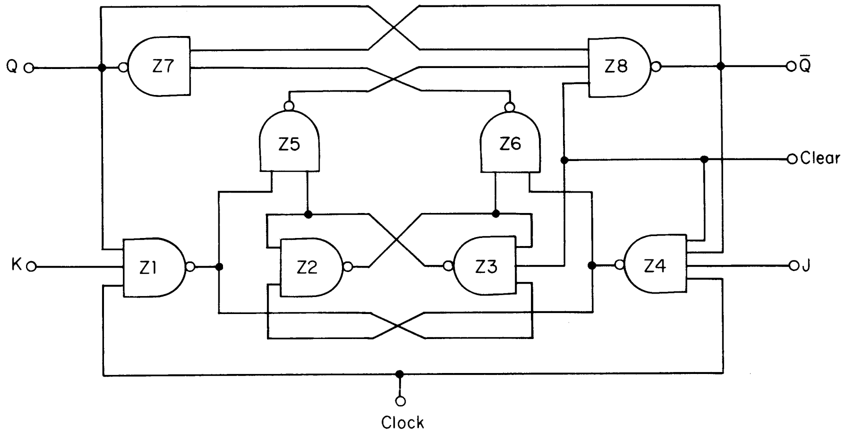
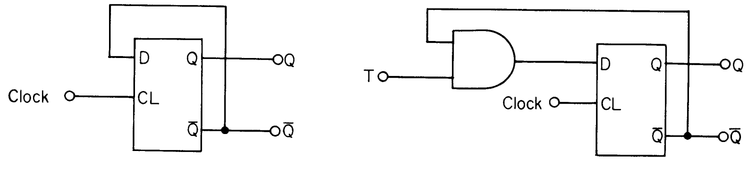


Fig. 7.13. Typical J-K master-slave flip-flop circuit.



(a) Toggle only

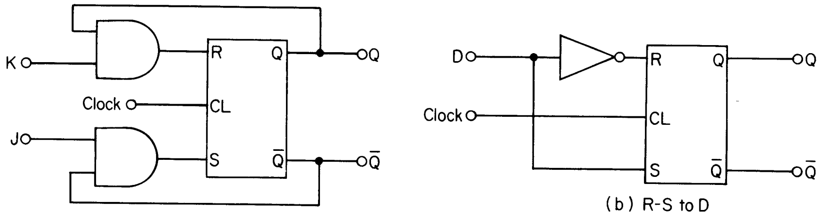
(b) With a T data input

Fig. 7.14. Converting *D* to *T* flip-flop: (*a*) toggle only; (*b*) with a *T* data input.

It is often desirable to interchange flip-flop types with little or no additional gating. Most types are interchangeable to some extent, as a comparison of truth tables suggests. Figure 7.14*a* shows the generation of the *T* type by simply connecting the $\overline{Q}$ output to the *D* input. This configuration is equivalent only to the *T* type with the *T* input at a logical 1. Figure 7.14*b* shows a *D*-to-*T* equivalent functionally identical to the *T* type.

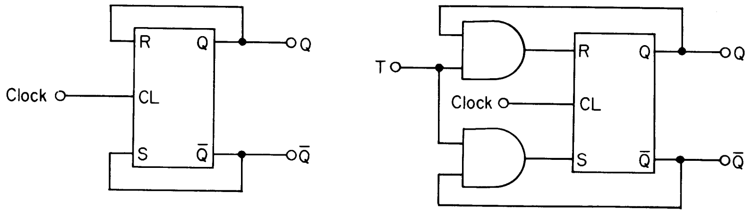
Figure 7.15 shows the *R-S* in *J-K, D,* and both *T* configurations. The *J-K* is implemented by gating the outputs back to the inputs, using two input AND gates. The *D* is made by using an inverter on the *R* input and connecting it to the *S* input. Of the two types of *T* conversions shown, one is equivalent to the *T* with the input always at a 1 level, and the other shows the complete *T* function using two AND gates.

The *J-K* type can be used to implement both *D*- and *T*-type functions quite easily. The *D* is accomplished by using an inverter between the *J* and *K* inputs as shown in Fig. 7.16. A *D* type is also shown which makes use of a *K** input available on some *J-K* types. The *K** input is merely the inverse of the *K* input or $\overline{K}$. The *T* type is formed by tying the *J* and *K* inputs together; it is not actually manufactured as a flip-flop but is implemented from one of the other types. The *D, R-S,* and *J-K* types are all manufactured in some configuration similar to those described.



(a) R-S to J-K

(b) R-S to D



(c) R-S to T (toggle only)

(d) R-S to T with T data input

Fig. 7.15. *R-S* to *J-K, D,* and *T* flip-flops.

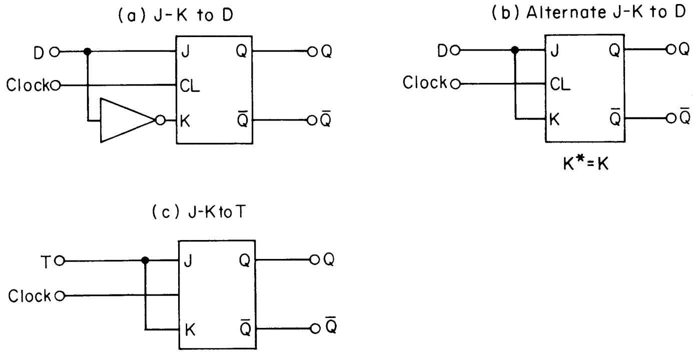


Fig. 7.16. *J-K* to *D* and *T* flip-flops.

7.2 SERIES 54/74 FLIP-FLOPS

The flip-flops discussed in this section are considered functionally identical in terms of internal circuit operation. Differences in asynchronous inputs and preset and clear, which will be pointed out within each group, are not to be considered as part of the synchronous function. The flip-flops indicated include all TTL devices, standard, high-speed (H), and low-power (L), and are listed in numerical order in each subgroup. Unless otherwise noted, the following statements are true for all flip-flops listed.

1. For preset and clear inputs:

   preset = 0, $Q = 1$
   clear = 0, $Q = 0$

PRESET	CLEAR	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	No change	

   These inputs will also be considered asynchronous, and independent of the clock. While they are present, all other operations are inhibited.
2. All flip-flops are d-c coupled.
3. Standard and H series include clamping diode on clock inputs.
4. For all flip-flops both Q and $\overline{Q}$ outputs are available.
5. Setup and hold times are measured from the clock triggering edge as mentioned earlier.
6. All devices are available in 14-pin or 16-pin flat or dual-in-line packages.
7. All devices are available in ceramic or plastic packages.

***J-K* Positive Edge-triggered.** The SN54/7470 have both preset and clear. Preset and clear functions should occur only while the clock is low, to prevent erroneous operation. This device has three each of the *J* and *K* inputs including a *J** and *K** (inverted *J* and *K*). The SN54/7470 are primarily useful in medium- to high-speed systems where gating is required on *J* and *K* inputs. This device does not have a clock clamping diode.

***J-K* Negative Edge-triggered.** Presently all negative edge-triggered *J-K*'s are in the H series and operate at clock rates of 50 MHz (typical). Operation is guaranteed for 40 MHz as long as minimum setup times are observed.

SN54/74H101 have *J* and *K* inputs consisting of 2-wide 3-input AND-OR gates. One input of each gate is connected to form the clock input. A single asynchronous input is available, preset. The SN54/74H101 are very useful in applications such as up/down counters and right-left shift registers when AND-OR gating is required for *J* and *K* inputs.

SN54/74H102 are the same as H101 except that *J* and *K* inputs are 4-input AND gates with two inputs tied together for the clock. Both preset and clear inputs are available. The SN54/74H102 are useful in synchronous counters to eliminate the need for external gates.

SN54/74H103 consist of two flip-flops in a single package with only one *J* and one *K* input for each. Separate clear and clock inputs are provided for each flip-flop. The SN54/74H103 are ideal for shift register applications.

SN54/74H106 are identical to the H103 with two exceptions: the SN54/74H106 are available in a 16-pin dual-in-line package only, and they have both preset and clear inputs for each flip-flop.

SN54/74H108 are the same as the H103 except that a common clock and a common clear input are used for both flip-flops. Separate preset inputs are available.

J-K Master-slave. These flip-flops operate from a complete clock pulse, and the outputs change on the negative transition. A positive clock pulse is used. As mentioned before, for error-free operation data on the *J* and *K* inputs should be stable while the clock is high.

SN54/74H71 are the master-slave equivalents of the H101 with AND-OR *J* and *K* inputs and a preset input.

SN54/7472, SN54/74H72, and SN54/74L72 have three each of the *J* and *K* AND gated inputs and both preset and clear inputs. As noted, they are available in all three series with identical pin configurations. These are the master-slave equivalents of the H102.

SN54/7473, SN54/74H73, and SN54/74L73 are master-slave equivalents of the H103. They are dual flip-flops with single *J* and *K* inputs and separate clock and clear inputs.

SN54/7476 and SN54/74H76 are master-slave equivalents of the H106. They are dual *J-K*'s with separate clock, preset, and clear inputs.

SN54/74H78 and SN54/74L78 are master-slave equivalents of the H108, dual flip-flops with common clock and clear inputs and separate preset inputs.

SN54/74104 have four *J* and *K* AND inputs and separate preset and clear inputs. One *J* input is tied to one *K* input to give a gate-type input (*J*-*K*). If this *J*-*K* input is a logical 0, flip-flop operation is inhibited. Since the clock input is buffered with an inverter, it is necessary to maintain the data inputs stable while the clock is low for this device. Capacitive delay is provided on the *J* and *K* inputs to increase the allowable clock skew. Maximum allowable delay at 25°C is about 20 ns, and the maximum clock frequency is about 30 MHz.

SN54/74105 are similar to the SN54/74104 described above with two exceptions: one each of the three remaining *J* and *K* inputs is inverted, and delaying capacitors are not included. As a result the SN54/74105 will toggle slightly faster than the SN54/74104.

SN54/74107 are functionally identical to the SN54/7473, but the pin configuration is different. The SN54/74107 are available only in the dual-in-line package.

R-S Master-slave. SN54/74L71 are similar to the L72. The only difference is *R-S* rather than *J-K* operation. This is advantageous in some applications where data inputs cannot be held stable while the clock is high. The devices operate correctly if minimum setup times are observed.

D Positive Edge-triggered. SN54/7474 and SN54/74H74 are dual flip-flops with single *D* inputs and separate preset and clear inputs. The devices are useful in buffer and storage registers, shift registers, and counters where little additional gating is required on the inputs.

D Latch. SN54/7475 have a single *D* input and a common clock for two flip-flops with four latches per package. The 54/7475 are primarily used in data storage applications.

SN54/74100 are functionally the same as the SN54/7475 except that the 54/74100 have eight latches with two clock inputs. Four latches have common clock inputs.

J-K Master-slave with Data Lockout. SN54/74110 have three *J* and *K* AND inputs with both preset and clear inputs. These flip-flops have data lockout capability which eliminates the need to hold the data inputs stable while the clock is high. It is only necessary to observe the setup and hold times with respect to the rising edge of the clock pulse to store the data present in the master section. After the rising edge of the clock pulse, the data inputs may change without affecting the outputs. The data thus stored in the master are transferred to the slave on the trailing edge of the clock. These flip-flops make the amount of allowable clock skew directly proportional to the width of the clock pulse. Therefore, the pulse width may be increased as desired to meet system requirements.

SN54/74111 are flip-flops with a clocking mechanism identical to the SN54/110. However, only single *J* and *K* inputs with separate clock, clear, and preset inputs are provided. The SN54/74111 are available in a 16-pin dual-in-line package only.

7.3 FLIP-FLOP APPLICATIONS

This section includes a variety of flip-flop circuits, primarily intended to demonstrate the extremely broad field of flip-flop application. Probably the most prominent use of flip-flops is in counter circuits, which are discussed in Chapter 10. Probably the next most familiar circuits using flip-flops are shift and storage registers. Figure 7.17 shows serial shift registers implemented with the *J-K, R-S,* and *D*-type flip-flops. In general these registers will have common clear and individual preset lines.

Figure 7.18 presents a 40-MHz serial shift register with asynchronous parallel-load capability. The register is implemented with SN74H106 flip-flops. The asynchronous parallel load is accomplished by using two SN74H00 gates per flip-flop. They are connected in such a way that a logical 1 on the load input enables the parallel inputs. When the load input returns to the logical 0, synchronous operation is resumed with the preset data loaded in. While the load input is at a logical 1, the serial operation is totally inhibited.

One other general type of shift register is the left-right shift. Figure 7.19 demonstrates a circuit of this type utilizing *D*-type flip-flops (SN7474s or H74s). The same function can be obtained by using either *R-S* or *J-K* types, but one additional inverter is needed at each flip-flop input to make *D*-type flip-flops from them.

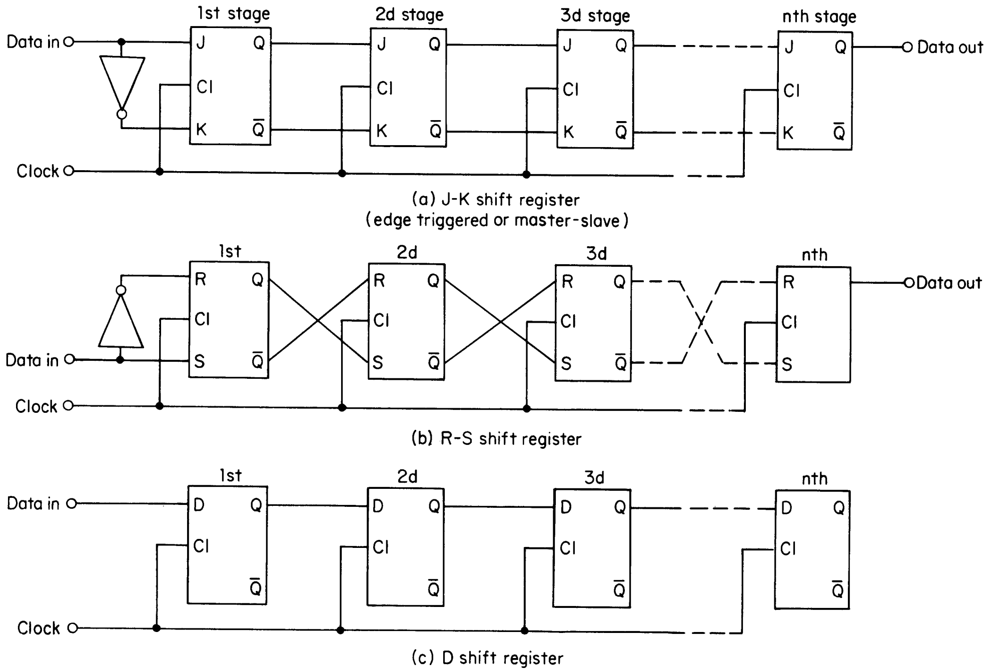

1st stage
2d stage
3d stage
nth stage
Data in
Data out
Clock
J
Q
Cl
K
$\bar{Q}$
(a) J-K shift register
(edge triggered or master-slave)
1st
2d
3d
nth
R
S
(b) R-S shift register
D
(c) D shift register

Fig. 7.17. Flip-flops used as shift registers.

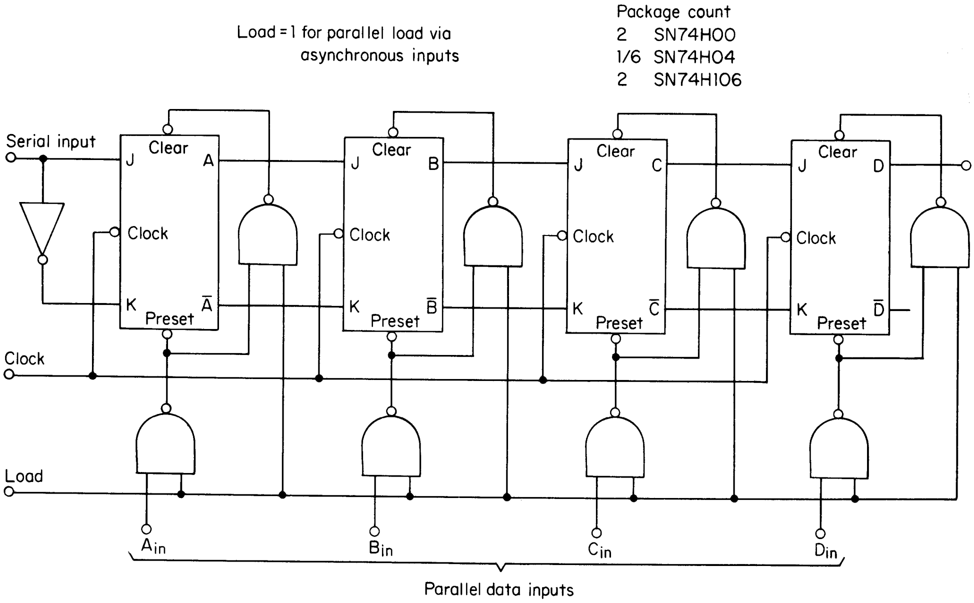

Load = 1 for parallel load via asynchronous inputs
Package count
2 SN74H00
1/6 SN74H04
2 SN74H106
Serial input
Clear
J
A
B
C
D
Clock
K
$\bar{A}$
$\bar{B}$
$\bar{C}$
$\bar{D}$
Preset
Load
A_{in}
B_{in}
C_{in}
D_{in}
Parallel data inputs

Fig. 7.18. 40-MHz shift register with asynchronous parallel load.

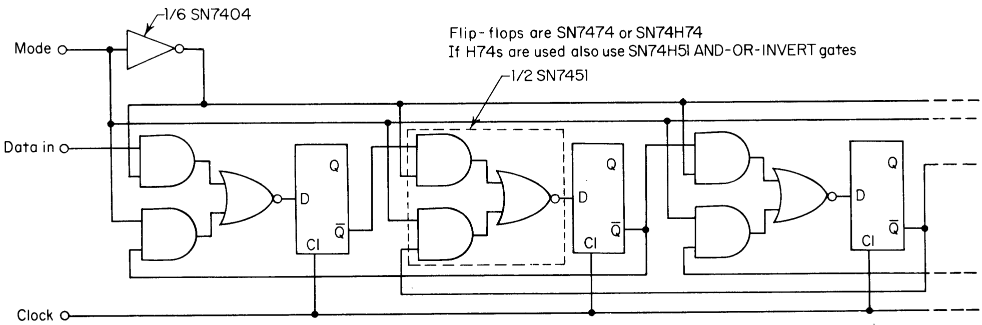
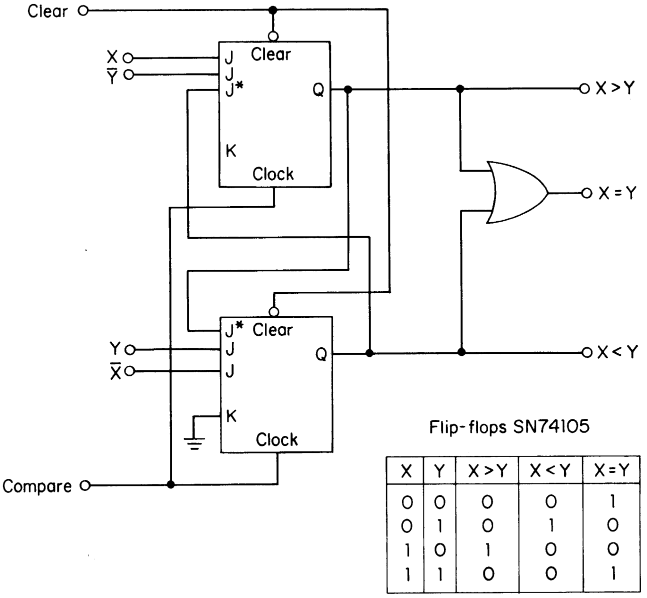

1/6 SN7404
Mode
Flip-flops are SN7474 or SN74H74
If H74s are used also use SN74H51 AND-OR-INVERT gates
1/2 SN7451
Data in
Q
D
Cl
$\bar{Q}$
Clock

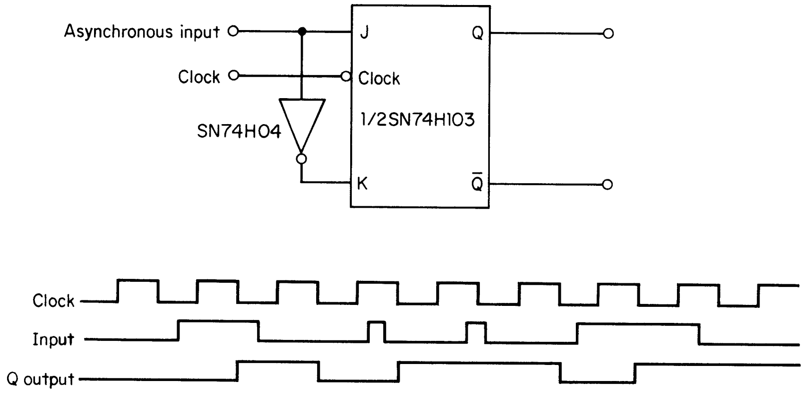
Fig. 7.19. Left-right shift register.

A wide variety of serial decode, comparison, and timing functions can be accomplished by using flip-flops. A serial comparator is shown in Fig. 7.20. The operation of the circuit is as follows: Initially a clear pulse is given, which sets both Q outputs to logical 0. All possible X and Y inputs prior to clocking the three outputs $X > Y$, $X < Y$, $X = Y$ are defined in the truth table after a compare pulse is applied. (Note: SN74105s are positive edge-triggered.) If two words A and B are serially applied to the comparator 2 bits at a time, the $X = Y$ output will be at a logical 1 until 2 bits of words A and B are unequal. At that time, depending on which one is greater, either the $X > Y$ or $X < Y$ output will be a logical 1. Both Q outputs are connected back to opposite J^* inputs, generating a latching effect when either $X > Y$ or $X < Y$ is logical 1; the outputs will remain in that state until another clear pulse is applied.

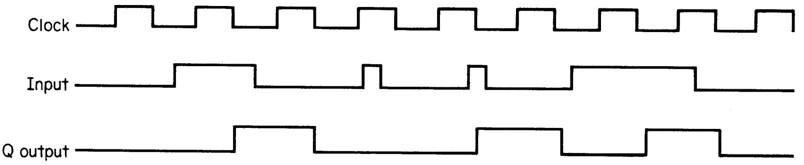

Clear
X
$\bar{Y}$
J
J
J*
Clear
Q
K
Clock
X>Y
X=Y
J*
Clear
Y
$\bar{X}$
J
J
Q
K
Clock
X<Y
Compare
Flip-flops SN74105

X	Y	X>Y	X<Y	X=Y
0	0	0	0	1
0	1	0	1	0
1	0	1	0	0
1	1	0	0	1

Fig. 7.20. Serial magnitude comparator with truth table.



Timing for J-K master-slave flip-flop, SN7473



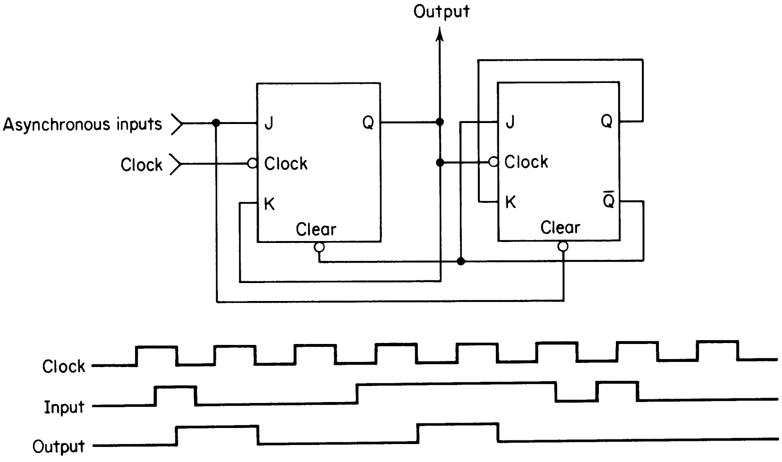
Timing for J-K negative edge triggered flip flop, SN74H103

Fig. 7.21. Synchronizer for asynchronous input.

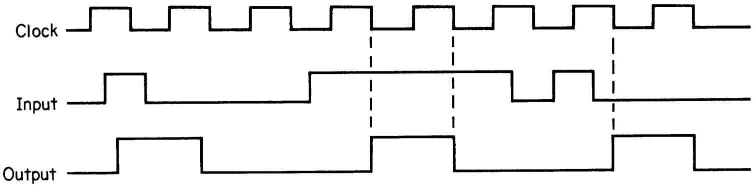
Many operations involve synchronizing data with respect to a system clock. Figure 7.21 demonstrates how this can be accomplished with both *J-K* edge-triggered and master-slave flip-flops. The difference is the fact that the master-slave will synchronize input data of any width greater than about 20 ns, which is present while the clock is high, but the edge-triggered type will synchronize data narrower than the clock pulse only if it is present for the setup time at the clock edge that activates the flip-flop.

Another circuit that synchronizes data and generates only a single output pulse always of the same width is shown in Fig. 7.22. In general the data-in must be at least as wide as the clock pulse or it is not recognized (unless it occurs at the clock trailing edge). Again this circuit can be implemented with *J-K* master-slave flip-flops, the only difference in operation being that the master-slave will recognize data of any width greater than about 20 ns that occur while the clock is high.

A synchronized clock-burst generator is shown in Fig. 7.23. Its *Q* output is a train of pulses sustained as long as the *asynchronous input* is at a logical 1. The *Q* output is positive-going pulses whose width is equal to the time the clock pulse is at logical 0 level. As in the other circuits using *J-K* master-slave flip-flops, this circuit recognizes any positive data regardless of width (>20 ns) while the clock is at a logical 1. This feature of the master-slave device makes this circuit useful as a 1's detector


Output
Asynchronous inputs
Clock
J
Q
Clock
K
Clear
J
Q
Clock
K
Q̄
Clear
Clock
Input
Output

Timing for J-K negative edge triggered flip-flop


Clock
Input
Output

Timing for J-K master-slave flip-flop

Fig. 7.22. Synchronized uniform-pulse-width "single-pulse" generator.

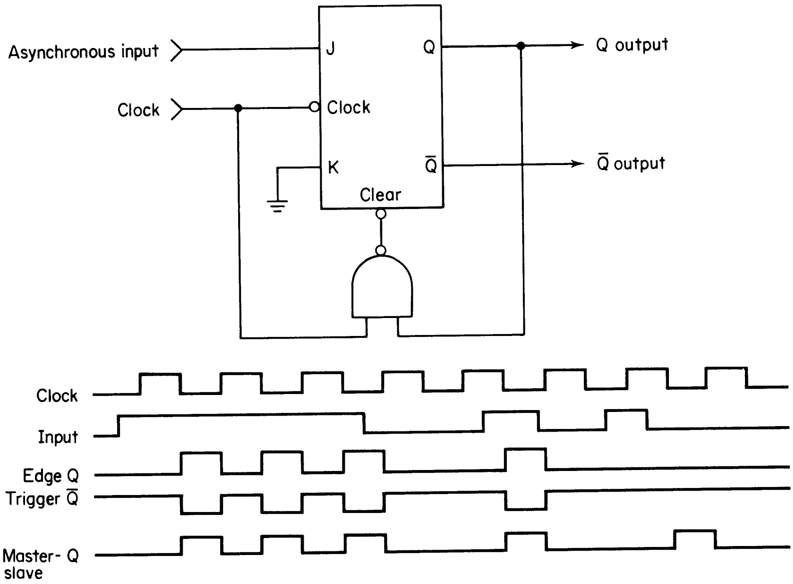

Asynchronous input
Clock
J
Q
Clock
K
Q̄
Clear
Q output
Q̄ output
Clock
Input
Edge Q
Trigger Q̄
Master- Q
slave

Fig. 7.23. Synchronized clock-burst generator.

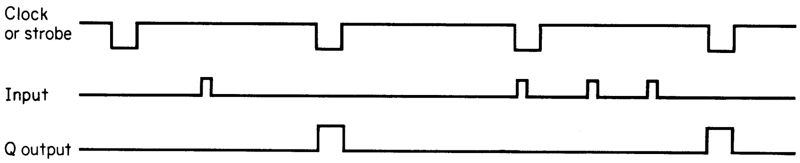

Clock or strobe
Input
Q output

Fig. 7.24. Timing for circuit of Fig. 7.23 as 1's detector.

or an error detector. The timing diagrams in Fig. 7.24 show the operation of the same circuit with a master-slave unit and with the clock operating as a strobe input. The operation is such that if a logical 1 level is ever present at the asynchronous input while the clock (or strobe) is high, the output will rise to a logical 1 during the clock (or strobe) 0 interval.

In many MOS systems a two-phase clock system is required. Figure 7.25 shows how the clock pulse is generated and translated to MOS voltage levels. The outputs will drive 400-pF loads with rise and fall times typically less than 40 ns. For high-threshold MOS it will be necessary to bias the transistors at $-V_G$ and use a resistor divider or a zener to provide the level translation at the bases of the transistors.

A variety of one-shots can be generated by using flip-flops in the configuration of Fig. 7.26. As indicated, for an even number of gate delays the $\overline{Q}$ output is connected through the gates to the clear input. These circuits, when implemented

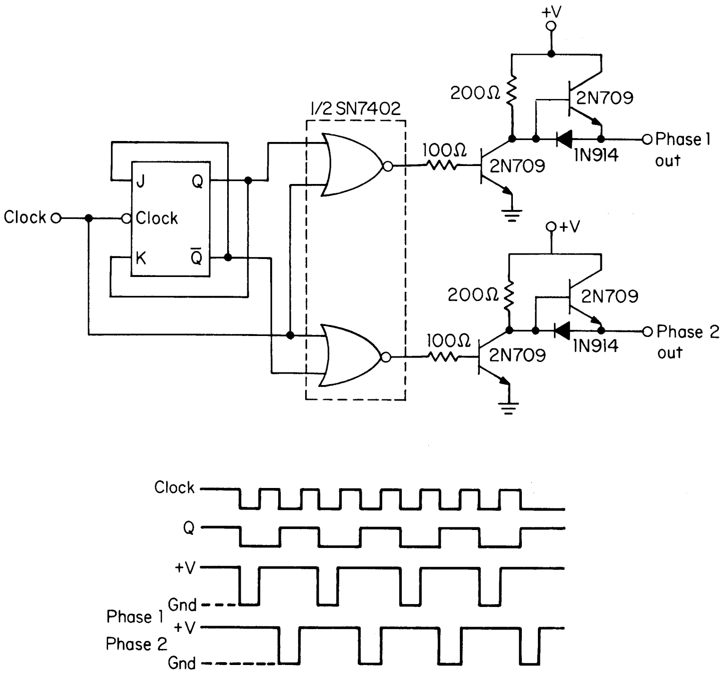
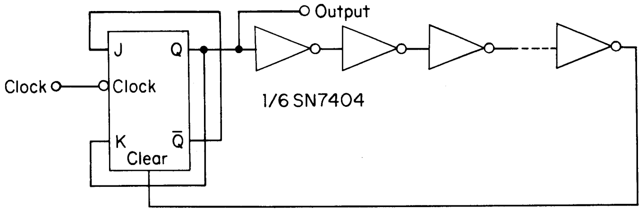

+V
200Ω
2N709
1/2 SN7402
100Ω
2N709
1N914
Phase 1 out
J
Q
Clock
Clock
K
Q̄
+V
200Ω
2N709
100Ω
2N709
1N914
Phase 2 out
Clock
Q
+V
Gnd
Phase 1
+V
Phase 2
Gnd

Fig. 7.25. MOS 2-phase clock generator.


Output
J
Q
Clock
Clock
K
$\bar{Q}$
Clear
1/6 SN7404

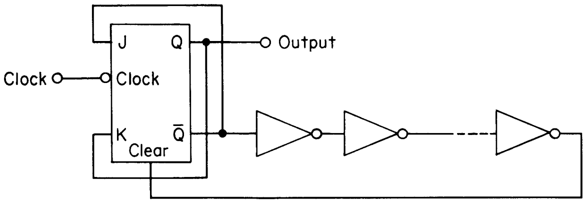
Output pulse width (typical) = T_{pd} F-F + T_{pd} gate (2n + 1); n = 0, 1, 2...

For SN7476 and n = 2

PW = 16 ns + 10 (5) ns

PW ≅ 66 ns

(a) For odd number of gate delays


Output
J
Q
Clock
Clock
K
$\bar{Q}$
Clear

PW = $T_{pd(ff)}$ + $T_{pd(gate)}$ (2n); n = 0, 1, 2...

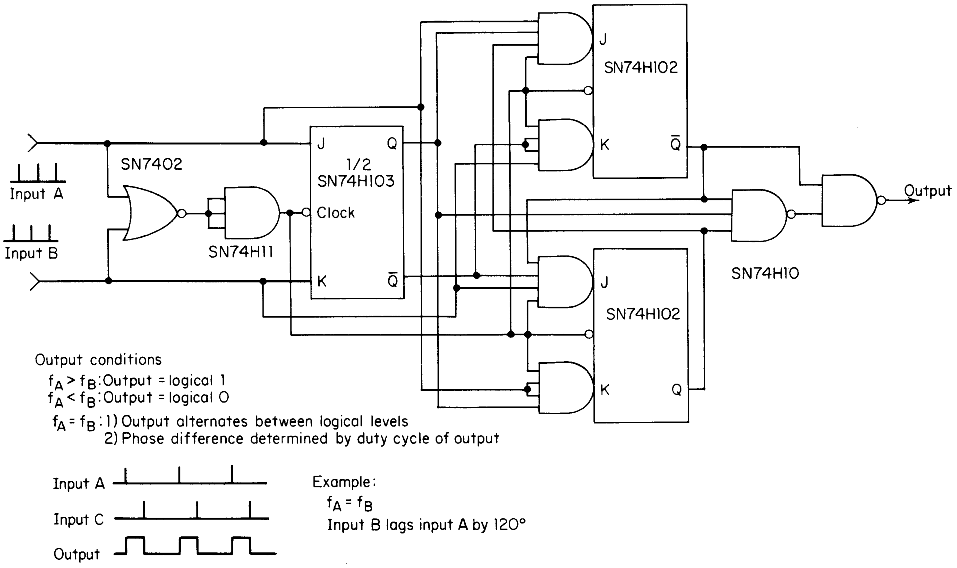
For SN7476 and n = 1

PW = 16 ns + 10 (2) ns

PW ≅ 36 ns

(b) For even number of gate delays

Fig. 7.26. Flip-flop one-shots.


Input A
Input B
SN7402
SN74H11
J
Q
1/2
SN74H103
Clock
K
$\bar{Q}$
J
SN74H102
K
$\bar{Q}$
J
SN74H102
K
Q
SN74H10
Output
Input A
Input C
Output

Output conditions

$f_A > f_B$: Output = logical 1

$f_A < f_B$: Output = logical 0

$f_A = f_B$: 1) Output alternates between logical levels

2) Phase difference determined by duty cycle of output

Example:

$f_A = f_B$

Input B lags input A by 120°

Fig. 7.27. Digital frequency-phase comparator using Series SN54/74H100 flip-flops.

with H series flip-flops, are capable of generating pulses as narrow as 12 ns typical with SN74H73. In practical applications the maximum pulse width should be about 250 ns, using low-power devices with six inverter delays.

Figure 7.27 shows a frequency-phase detector. The output indicates frequency differences, as well as phase difference when the frequencies are equal. The output can be measured with respect to either the A or B input. In using this circuit, one ambiguous condition arises. When A and B are 180° out of phase, the output will be a 50 percent duty cycle; when A and B are exactly in phase, the output will be a 50 percent duty cycle, but at *half the frequency* of the 180° condition. This condition is ambiguous only in the sense that if the output is to be integrated, the 0° and 180° conditions will generate identical results.

8

Decoders

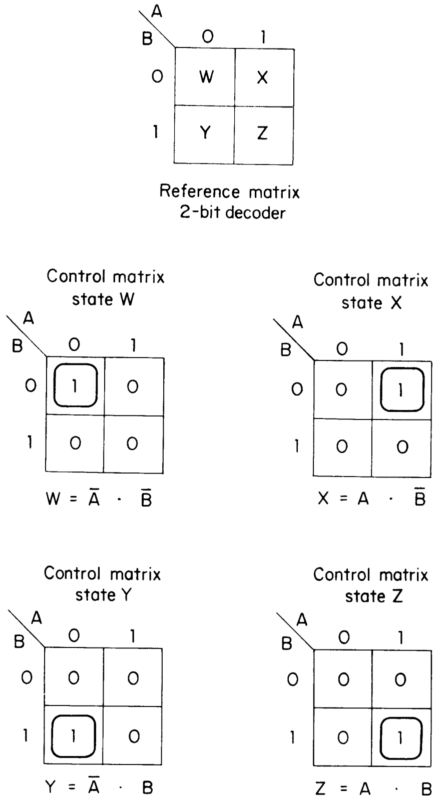
Many systems require the decoding of information represented by digital-machine-language words. Decoding is necessary in such applications as data multiplexing, rate multiplying, digital display, digital-to-analog converters, and memory addressing. It is accomplished by matrix systems that can be constructed from such devices as magnetic cores, diodes, resistors, transistors, and FETs. The most common decoding requirements can now be satisfied with off-the-shelf integrated circuits.

8.1 DECODER THEORY

A single binary word N digits in length can represent 2^N different elements of information. For example, the subset represented by a 2-bit binary word contains four such elements, each element representing one of the binary states (00, 01, 11, 10) of the word. In Fig. 8.1 each binary state is assigned to one of four letters (W, X, Y, Z). These letters correspond to the four outputs of a decoder with two inputs A and B. Karnaugh maps are then drawn for each state assignment. The map for W shows that output W is logical 1 for input bits A and B both equal to logical 0. The Boolean expression for output W is $W = \overline{AB}$. This same reasoning is repeated for the decoding of outputs X, Y, and Z. The equations are implemented by using 2-input NAND gates in Fig. 8.2.

Figure 8.3 shows the reference matrix for decoding a binary word of 3 bits. For simple decoding it is usually not necessary to draw control matrices for each state as was done in Fig. 8.1. Q is true, logical 1, for only $A = B = C = 0$. Since only Q is assigned to this state, $Q = \overline{ABC}$ can be directly written by inspection of the reference matrix. A 3-bit binary decoder whose control equations are given in Fig. 8.3 is implemented in Fig. 8.4.

Many machine languages do not use the binary number system to its full capacity. For example, the common BCD (binary-coded decimal) code uses only 10 of the possible 16 states of a 4-bit word. For this reason, a variety of codes can be generated for binary representation of decimal numbers. Figure 8.5 shows some of the common codes. Different codes have been generated to facilitate addition



B \ A	0	1
0	W	X
1	Y	Z

Reference matrix
2-bit decoder

Control matrix
state W

B \ A	0	1
0	1	0
1	0	0

$W = \bar{A} \cdot \bar{B}$

Control matrix
state X

B \ A	0	1
0	0	1
1	0	0

$X = A \cdot \bar{B}$

Control matrix
state Y

B \ A	0	1
0	0	0
1	1	0

$Y = \bar{A} \cdot B$

Control matrix
state Z

B \ A	0	1
0	0	0
1	0	1

$Z = A \cdot B$

Fig. 8.1. Control matrices for 2-bit decoder.

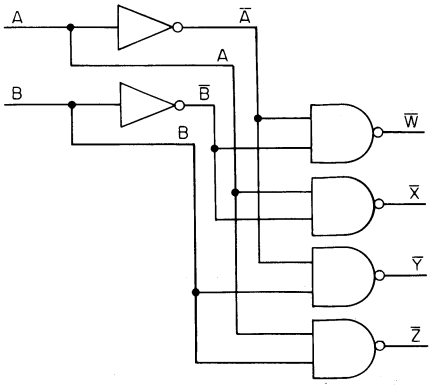
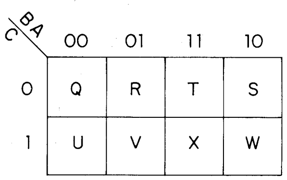


Fig. 8.2. 2-line-to-4-line decoder.



C \ BA	00	01	11	10
0	Q	R	T	S
1	U	V	X	W

$Q = \bar{A} \cdot \bar{B} \cdot \bar{C}$
$R = A \cdot \bar{B} \cdot \bar{C}$
$S = \bar{A} \cdot B \cdot \bar{C}$
$T = A \cdot B \cdot \bar{C}$
$U = \bar{A} \cdot \bar{B} \cdot C$
$V = A \cdot \bar{B} \cdot C$
$W = \bar{A} \cdot B \cdot C$
$X = A \cdot B \cdot C$

Fig. 8.3. Reference matrix for 3-bit decoder.

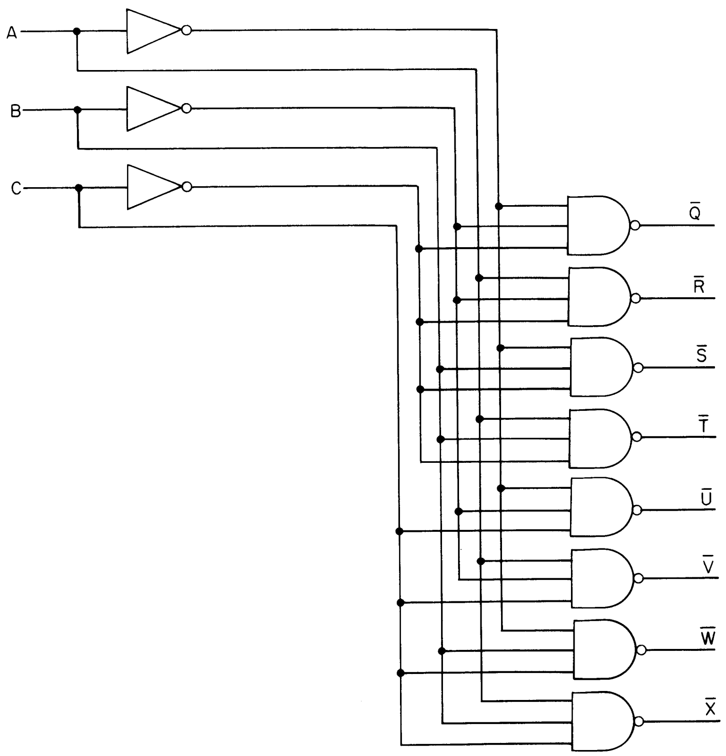

A
B
C
$\overline{Q}$
$\overline{R}$
$\overline{S}$
$\overline{T}$
$\overline{U}$
$\overline{V}$
$\overline{W}$
$\overline{X}$

Fig. 8.4. 3-line-to-8-line decoder.

Binary number	8-4-2-1 or BCD	2-4-2-1	Excess-3	Excess-3 Gray
0000	0	0		
0001	1	1		
0010	2	2		0
0011	3	3	0	
0100	4	4	1	4
0101	5		2	3
0110	6		3	1
0111	7		4	2
1000	8		5	
1001	9		6	
1010			7	9
1011		5	8	
1100		6	9	5
1101		7		6
1110		8		8
1111		9		7

Fig. 8.5. 4-bit binary codes for decimal representation.

Decimal digit	5-1-1-1	8-6-4-2-1	Unweighted
0	00000	00000	00000
1	00001	00001	10000
2	00011	00010	11000
3	00111	00011	11100
4	01111	00100	11110
5	10000	00101	11111
6	11000	01000	01111
7	11100	01001	00111
8	11110	10000	00011
9	11111	10001	00001

Fig. 8.6. More-than-4-bit coder for decimal representation.

and subtraction, or complementing, or decoding to decimal equivalent, or error detection and correction.

The first two codes shown in Fig. 8.5 (8-4-2-1 or BCD, and 2-4-2-1) are called *weighted codes.* In a weighted code, direct mathematical conversion can be made from the binary word to its decimal equivalent. For example:

Code weights = 8-4-2-1
Binary word = 1 0 0 1 (least significant bit, or LSB, rightmost)
Decimal equivalent = $(8 \cdot 1) + (4 \cdot 0) + (2 \cdot 0) + (1 \cdot 1) = 9$

The excess-3 code shown in Fig. 8.5 is a biased weighted code. Conversion from an excess-3 code to its decimal equivalent can be done in the following manner:

Reference matrix

DC \ BA	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X	X	X	X
10	8	9	X	X

Control equations

$0 = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$
$1 = A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$
$2 = \bar{A} \cdot B \cdot \bar{C} \cdot \bar{D}$
$3 = A \cdot B \cdot \bar{C} \cdot \bar{D}$
$4 = \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D}$
$5 = A \cdot \bar{B} \cdot C \cdot D$
$6 = \bar{A} \cdot B \cdot C \cdot \bar{D}$
$7 = A \cdot B \cdot C \cdot \bar{D}$
$8 = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D$
$9 = A \cdot \bar{B} \cdot \bar{C} \cdot D$

Fig. 8.7. BCD-to-decimal decoder with false-data rejection (matrix and equations).

Code weights = 8-4-2-1
Binary word = 1 1 0 0 (least significant bit, or LSB, rightmost)
Decimal equivalent = $(8 \cdot 1) + (4 \cdot 1) + (2 \cdot 0) + (1 \cdot 0) - 3 = 9$

For determining equivalent decimal representation of unweighted codes—for example, excess-3 Gray—some type of reference matrix showing state assignments must be known.

Any number of binary bits greater than 4 can also be used to represent decimal numbers. Some different codes are shown in Fig. 8.6. Note that the unweighted code shown is the counting sequence of a twisted-ring Johnson counter.

Figure 8.7 gives the reference matrix and derived Boolean equations for a BCD-to-decimal decoder. These equations are implemented in Fig. 8.8. States 1010,

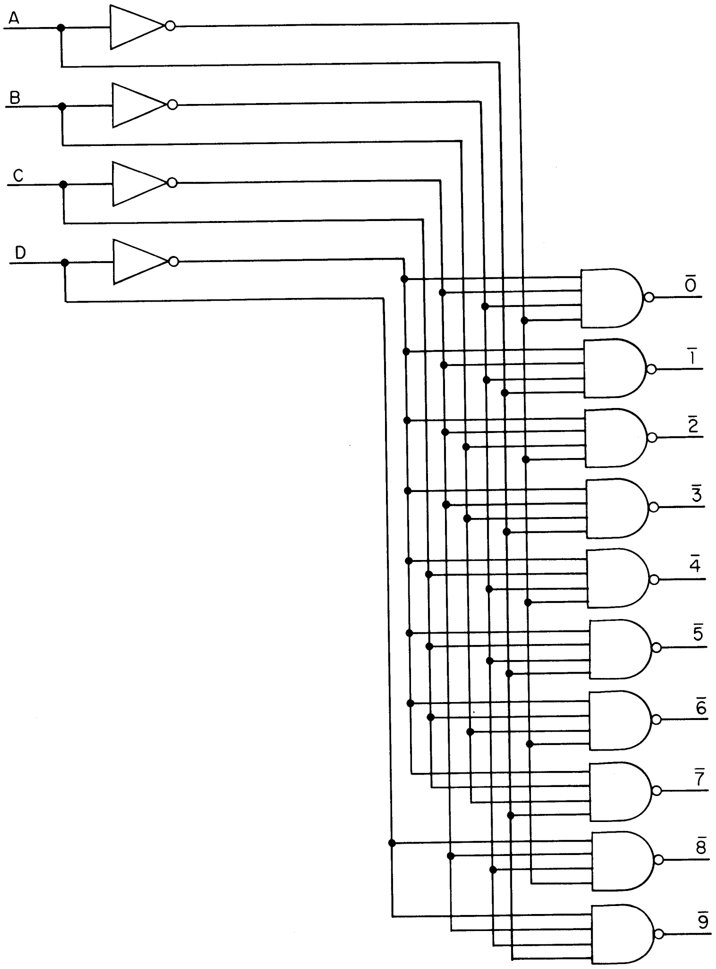

A
B
C
D
$\overline{0}$
$\overline{1}$
$\overline{2}$
$\overline{3}$
$\overline{4}$
$\overline{5}$
$\overline{6}$
$\overline{7}$
$\overline{8}$
$\overline{9}$

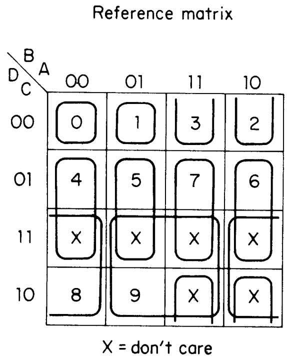
Fig. 8.8. BCD-to-decimal decoder with false-data rejection (logic diagram).

1011, 1100, 1101, 1110, and 1111 are not included in the 8-4-2-1 BCD code and are therefore considered to be false input data. Since all states are explicitly decoded, false data imposed on the inputs of this decoder result in all outputs being false, logical 1.

An 8-4-2-1 BCD-to-decimal decoder that does not reject false input data can also be built. This decoder minimizes the fan-in requirements of the NAND gates. The reference matrix and derived Boolean equations are shown in Fig. 8.9. *X* is entered in the unused cells of the reference matrix to indicate that the state of any defined output is "don't care" for false input data. For example, if 1111 is entered into this decoder, outputs 7 and 9 are both true. The decoder is implemented in Fig. 8.10.

In many applications it is desired that decoding of input states be done only during specific time intervals. Thus, any data that occur at the decoder inputs during certain intervals are rejected. This rejection is accomplished by adding a strobe input to the decoding matrix. When the strobe input is true, decoding is performed. When the strobe input is false, decoding is inhibited. Different strobing techniques must be used, depending on whether or not the decoding matrix rejects false data. Fig. 8.11 shows how a decode matrix that does reject false data may be strobed. This method actually enters false data (1111) into the decoder. The decode matrix rejects this input and all outputs remain false.

Another method that can be used to inhibit a decoding matrix requires an additional input on each of the decoder's NAND gates (see Fig. 8.12). All 10 of these



Control equations

$0 = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$
$1 = A \cdot \bar{B} \cdot \bar{C} \cdot \bar{D}$
$2 = \bar{A} \cdot B \cdot \bar{C}$
$3 = A \cdot B \cdot \bar{C}$
$4 = \bar{A} \cdot \bar{B} \cdot C$
$5 = A \cdot \bar{B} \cdot C$
$6 = \bar{A} \cdot B \cdot C$
$7 = A \cdot B \cdot C$
$8 = \bar{A} \cdot D$
$9 = A \cdot D$

Fig. 8.9. BCD-to-decimal decoder without false-data rejection (matrix and equations).

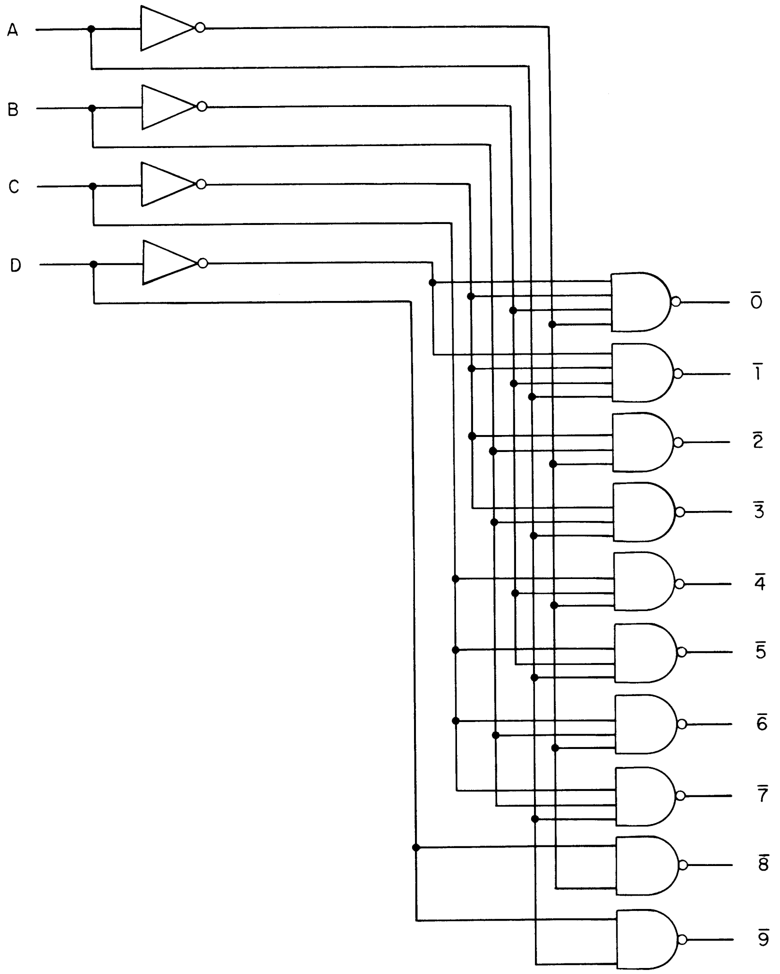

A
B
C
D
$\overline{0}$
$\overline{1}$
$\overline{2}$
$\overline{3}$
$\overline{4}$
$\overline{5}$
$\overline{6}$
$\overline{7}$
$\overline{8}$
$\overline{9}$

Fig. 8.10. BCD-to-decimal decoder without false-data rejection (logic diagram).

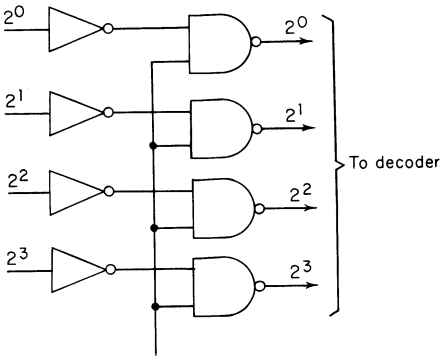

2^0
2^1
2^2
2^3
2^0
2^1
2^2
2^3
To decoder
Strobe

Fig. 8.11. Strobe circuit for BCD-to-decimal decoder with false-data rejection.

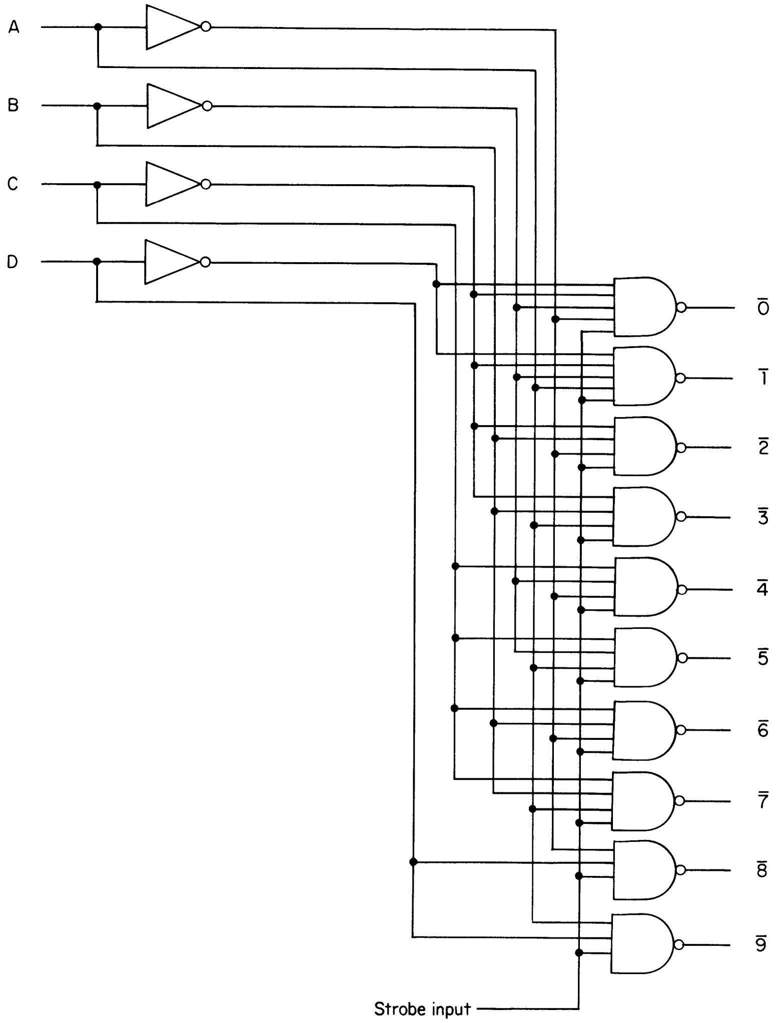

A
B
C
D
$\overline{0}$
$\overline{1}$
$\overline{2}$
$\overline{3}$
$\overline{4}$
$\overline{5}$
$\overline{6}$
$\overline{7}$
$\overline{8}$
$\overline{9}$
Strobe input

Fig. 8.12. BCD-to-decimal decoder with strobe input.

extra inputs are tied together. When these inputs are at logical 0, all decoder outputs are false. When strobe inputs are at logical 1, the decode matrix is enabled. This method of strobing works whether or not the decoder rejects false data.

Another method of strobing a decoding matrix uses latches as shown in Fig. 8.13. An advantage of this method is that decoded outputs are continuously available. Hence, if these decoded outputs are used to drive display devices, the display outputs do not blink. This strobing method also functions whether or not the decoder rejects false data. The outputs change state only while the strobe input is changing from 0 to 1.

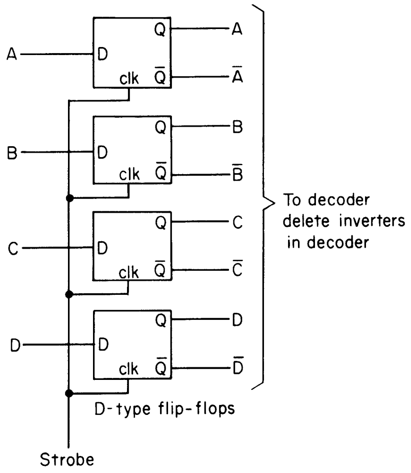


Fig. 8.13. Strobe circuit for BCD-to-decimal decoder without false-data rejection.

The twisted-ring Johnson counter uses five binary bits for its BCD count. A logic diagram, reference matrix, and decoder for this type of counter are shown in Fig. 8.14. This decoder does not reject false data. If a decoder with false-data rejection were built, five input NAND gates would be required, thus changing IC package count from $2\frac{1}{4}$ to 9.

Another special type of 8-4-2-1 BCD decoder which is commonly seen drives 7-segment display tubes. Since these displays require low voltage and current, allowing them to be driven directly from integrated circuitry, they are becoming widely popular. Figure 8.15 shows the segment layout of a typical display tube, and a matrix that shows how elements are controlled to generate digits. Note that tube elements are initially considered to be on. To produce specific digits, various display elements are turned off. This type of control requires fewer logic operations than if all tube elements were initially assumed to be off and were turned on to generate digits. Figure 8.16 shows the Karnaugh maps for control of each tube element. For this example it is assumed that the designed decoder need not reject false data, and therefore *X* ("don't care") is entered in the unused binary states. The derived control equations are implemented in Fig. 8.17.

8.2 SERIES 54/74 DECODERS AND DECODER/DRIVERS

Decoders in various configurations are available in Series 54/74 logic. A summary of these devices appears in Table 8.1. All these devices are of course basically decoders, but the additional terms used in the device descriptions require explanation here.

In the open-collector group, the word *driver* often appears in the description. This term implies that the output stage has special properties that make it useful in driving not only the usual logic gate input but other devices as well. Devices that can be driven include lamps, special displays, relays, and discrete transistors.

State table

Decimal	Counter A	B	C	Outputs D	E
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

Karnaugh map for decoding logic

ED \ CBA	000	001	011	010	110	111	101	100
00	0	1	2	X	X	3	X	X
01	X	X	X	X	X	4	X	X
11	8	X	X	X	6	5	X	7
10	9	X	X	X	X	X	X	X

X = don't care

$0 = \bar{A}\ \bar{E}$
$1 = A\ \bar{B}$
$2 = B\ \bar{C}$
$3 = C\ \bar{D}$
$4 = D\ \bar{E}$
$5 = A\ E$
$6 = \bar{A}\ B$
$7 = \bar{B}\ C$
$8 = \bar{C}\ D$
$9 = \bar{D}\ E$

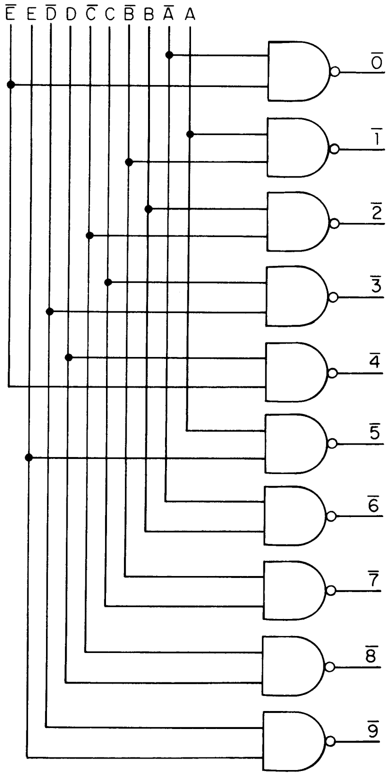
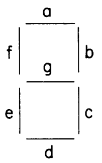


Fig. 8.14. Decoding of twisted-ring Johnson counter states, without false-data rejection.



Segment control

	a	b	c	d	e	f	g
0							☒
1	☒			☒	☒	☒	☒
2			☒			☒	
3					☒	☒	
4	☒			☒	☒		
5		☒			☒		
6	☒	☒					
7				☒	☒	☒	☒
8							
9				☒	☒		

☒ = segment off

Fig. 8.15. 7-segment display drive requirements.

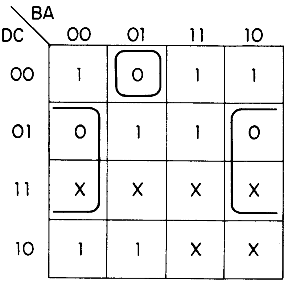
Another term that appears is *demultiplexer*. This means that additional gate inputs have been provided for data or strobe input or both.

A point sometimes overlooked with the open-collector drivers is that the maximum rated voltage of the output transistors is with supply voltage V_{CC} (nominal, of 5 V) applied to the device.

The breakdown voltage will drop considerably if V_{CC} is not applied. The reason for this is that with the output transistor in the off state and with V_{CC} applied, there is a saturated transistor across the base-emitter of the output transistor. This gives the effect of $V_{(BR)CER}$—collector-to-emitter breakdown voltage with a relatively low resistance from base to emitter. With no V_{CC} applied, the effect is that of $V_{(BR)CEO}$—breakdown for an open circuit from base to emitter, which is always considerably lower than $V_{(BR)CER}$. The requirement this characteristic imposes on system design is that V_{CC} should be applied before the high voltage on the output transistors is applied. If it is practical to reduce the high voltage while V_{CC} is off, then a safe value would be about one-half the normal value. The breakdown due to high voltage on the output transistors and V_{CC} off can result in permanent damage to the device.

Another situation occasionally overlooked in a system design is that voltage transients may be present on the voltage supply to the load that the driver output transistors are switching.

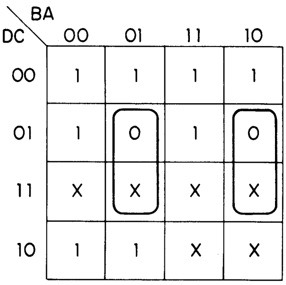
Segment a



DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	X	X	X	X
10	1	1	X	X

$\overline{a} = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot C$

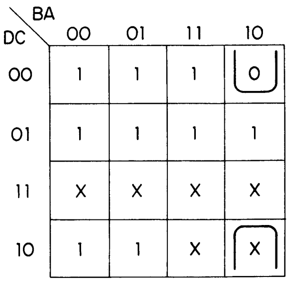
Segment b



DC \ BA	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

$\overline{b} = A \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot C$

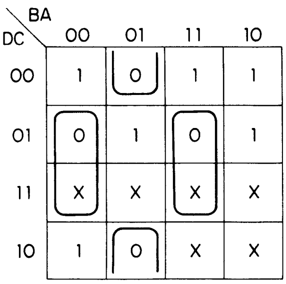
Segment c



DC \ BA	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$\overline{c} = \overline{A} \cdot B \cdot \overline{C}$

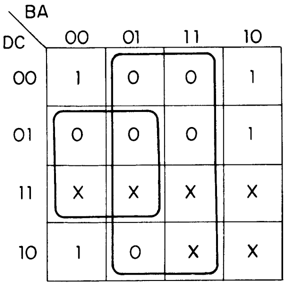
Segment d



DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	0	X	X

$\overline{d} = \overline{A} \cdot \overline{B} \cdot C + A \cdot B \cdot C + A \cdot \overline{B} \cdot \overline{C}$

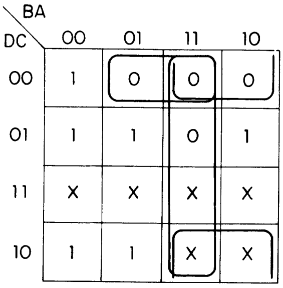
Segment e



DC \ BA	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

$\overline{e} = A + \overline{B} \cdot C$

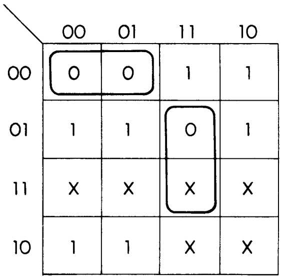
Segment f



DC \ BA	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

$\overline{f} = A \cdot \overline{D} \cdot \overline{C} + B \cdot A + B \cdot \overline{C}$

Segment g



	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

$\overline{g} = \overline{B} \cdot \overline{D} \cdot \overline{C} + B \cdot A \cdot C$

X = don't care

Fig. 8.16. Segment control matrices; 1 = segment on.

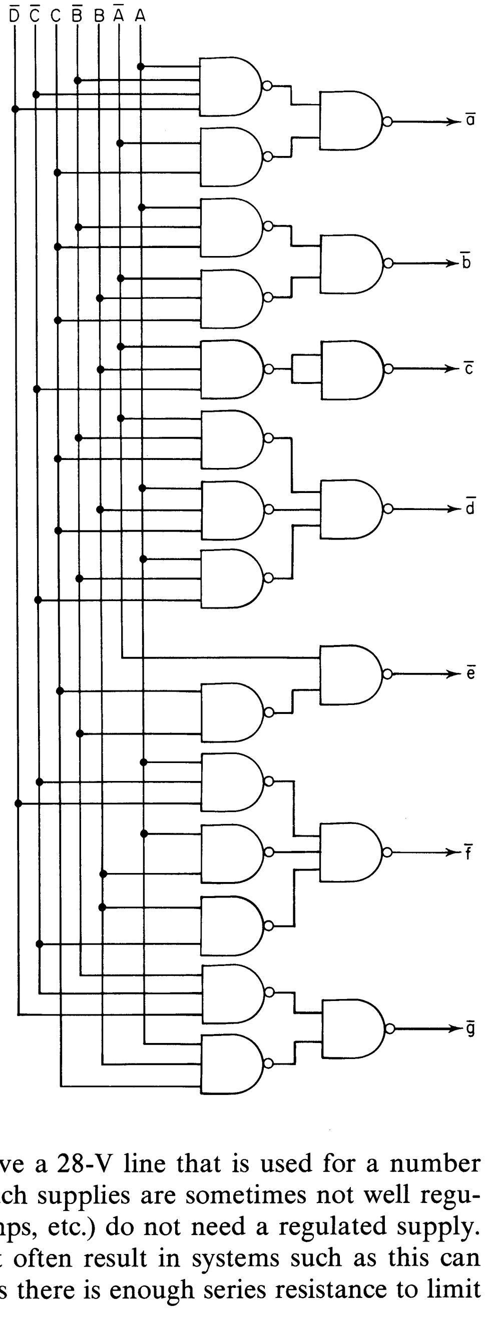


Fig. 8.17. BCD-to-7-segment decoder without false-data rejection.

It is quite common, for example, to have a 28-V line that is used for a number of control-type functions in a system. Such supplies are sometimes not well regulated, because the usual loads (relays, lamps, etc.) do not need a regulated supply. Unfortunately, the voltage transients that often result in systems such as this can destroy the driver output transistors unless there is enough series resistance to limit the current to a safe value.

BCD-to-decimal Decoder/Driver for Cold-cathode Tubes. The SN54/74141 (Fig. 8.18) is an 8-4-2-1 BCD-to-decimal decoder designed specifically to drive cold-cathode indicator tubes such as the Burroughs Nixie® or Raytheon Datavue® display tubes. This decoder minimizes switching transients and thereby maintains

Table 8.1. Summary of 54/74 Decoders and Decoder/Drivers

Open-collector outputs

Type	Description	Typical power dissipation, mW	Packages available	Zero-blank provisions	I_{OH} (μA) max at V_{OH}	V_{OH}, V	V_{OL} (volts) max at I_{OL}	I_{OL}, mA
SN54/7445	BCD-to-decimal decoder/driver	215	J, N	N/A	250	30	0.9	80
SN54/7446	BCD-to-7-segment decoder/driver	265	J, N	Yes	250	30	0.4	20
SN54/7447	BCD-to-7-segment decoder/driver	265	J, N	Yes	250	15	0.4	20
SN54/7448	BCD-to-7-segment decoder/driver	265	J, N	Yes	*	. . .	0.4	6.4
SN54/7449	BCD-to-7-segment decoder/driver	165	S	Yes	250	5.5	0.4	10
SN54/74141	BCD-to-decimal decoder/driver (cold-cathode tubes)	55	J, N	Yes	50	55	2.5	7
SN54/74145	BCD-to-decimal decoder/driver	215	J, N	N/A	250	15	0.9	80
SN54/74156	Dual 2-line-to-4-line decoder/demultiplexer	125	J, N	N/A	250	5.5	0.4	16

Standard totem-pole outputs

Type	Description	Typical power dissipation, mW	Packages available
SN54/7442	BCD-to-decimal decoder	140	J, N
SN54/7443	Excess-3-to-decimal decoder	140	J, N
SN54/7444	Excess-3-Gray-to-decimal decoder	140	J, N
SN54/74154	4-line-to-16-line decoder/demultiplexer	170	N
SN54/74155	Dual 2-line-to-4-line decoder/demultiplexer	125	J, N

*DTL-type output: 2-kΩ pull-up resistor.

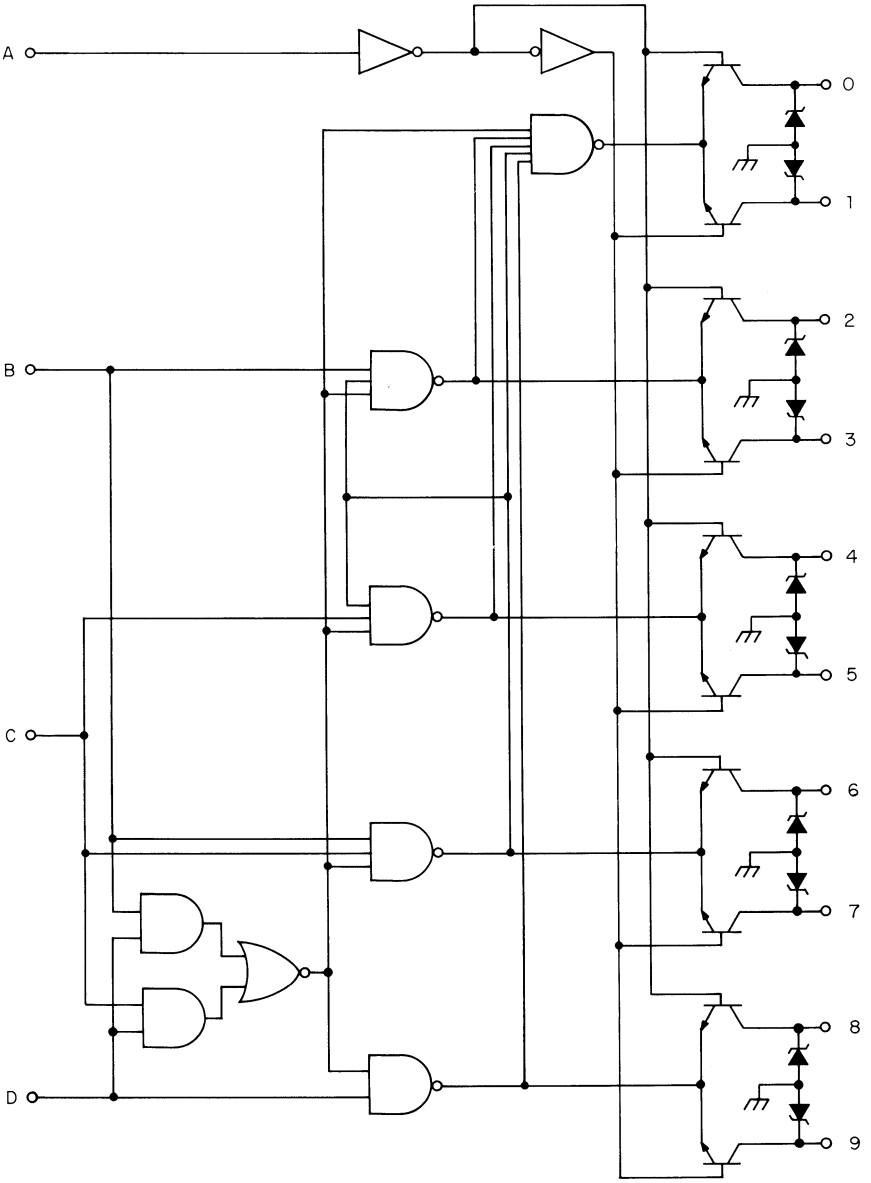


Fig. 8.18. BCD-to-decimal decoder/driver, SN54/74141.

a stable display. Full decoding is provided for all possible input states. For inputs of 10 through 15, all the outputs are off. With this feature and some additional external circuitry, the invalid codes can be used to obtain blanking of leading zeros or trailing zeros.

Figure 8.19 shows a two-decade display using a suggested method for blanking leading zeros. Any BCD input above decimal 9 may be used for blanking.

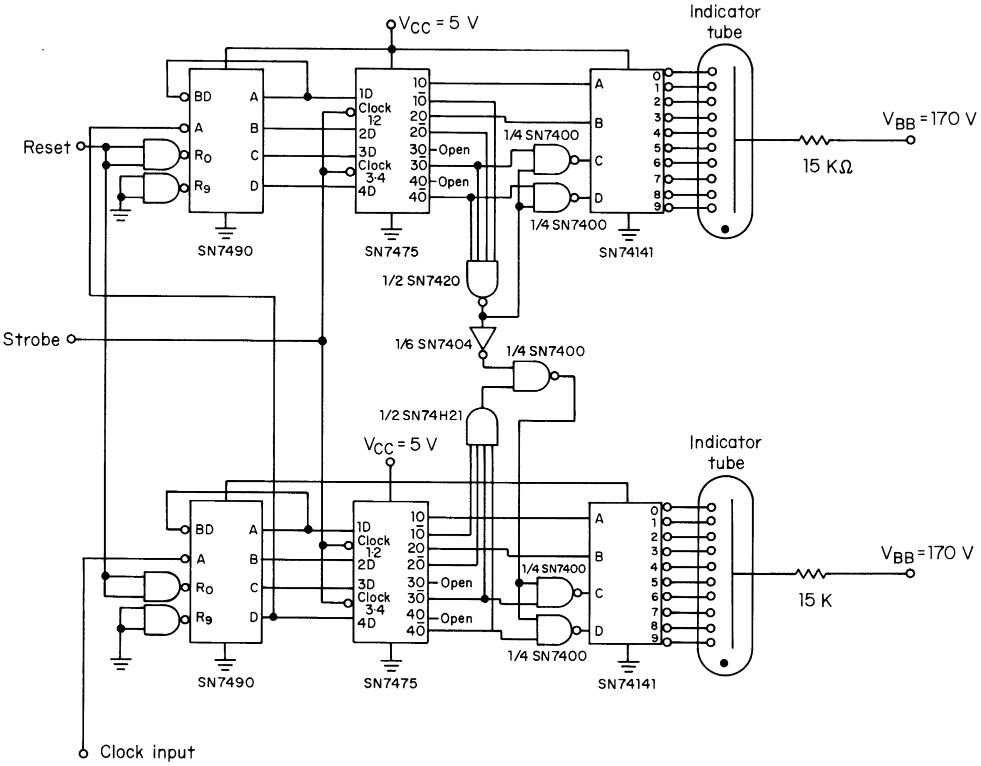

$V_{CC} = 5$ V
Indicator tube
Reset
Strobe
BD
A
R_0
R_9
A
B
C
D
SN7490
1D
Clock 1·2
2D
3D
Clock 3·4
4D
1Q
$\overline{1Q}$
2Q
$\overline{2Q}$
3Q
$\overline{3Q}$
4Q
$\overline{4Q}$
Open
Open
SN7475
1/4 SN7400
1/4 SN7400
SN74141
0
1
2
3
4
5
6
7
8
9
$V_{BB} = 170$ V
15 KΩ
1/2 SN7420
1/6 SN7404
1/4 SN7400
1/2 SN74H21
$V_{CC} = 5$ V
Indicator tube
SN7490
SN7475
1/4 SN7400
1/4 SN7400
SN74141
$V_{BB} = 170$ V
15 K
Clock input

Fig. 8.19. Typical application for SN54/74141 using zero-blanking feature (zero-to-99 counter and display).

When the most significant digit (MSD) is zero, it is blanked, and when both the MSD and least significant digit (LSD) are both zero, both digits are blanked. It may be desirable not to blank the zero LSD, because in such a case the whole display is blanked when the number is all zeros. To avoid blanking the zero LSD, the connection from the output of the SN7420 to the input of the SN7404 is broken, and the gating below this point is eliminated—the SN7404, SN74H21, and SN7400.

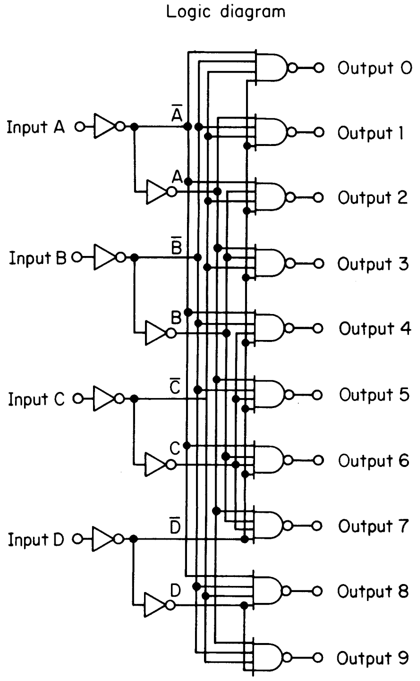
The output clamping zener diodes (Fig. 8.18) have a guaranteed minimum zener voltage of 60 V. This means that a swing of from 0 to 60 V may occur on the cathodes of the display tube in the blanking mode. A typical cold-cathode display tube is guaranteed to "fire" with 170 V across the tube, and a lower safe value of positive cathode voltage for negligible glow with $V_{BB} = 170$ V is about 50 V. Considering the operating specifications for display tube and driver, the biasing values shown in Fig. 8.19 for the display tube are suggested typical values.

Other BCD-to-decimal Decoder/Drivers. SN54/7442, SN54/7445, and SN54/74145 are 8-4-2-1 BCD-to-decimal decoders consisting of eight inverters and ten 4-input NAND gates (see Fig. 8.20). The inverters are connected in pairs to make BCD input data available for decoding by the NAND gates and to present only one standard 54/74 TTL load at each input. Full decoding of input logic assures that all outputs remain off for any false input conditions.

SN54/7442 has a standard TTL output configuration with fan-out capability of 10. SN54/7445 and SN54/74145 have open-collector outputs instead; each output is a high-performance n-p-n transistor designed for use as an indicator/relay driver or as an open-collector logic-circuit driver. Each of the high-breakdown output transistors (SN54/7445: V_{CE} of 30 V; SN54/74145: V_{CE} of 15 V) will sink up to 80 mA.

Other 4-line-to-10-line Decoders. SN54/7443 and SN54/7444 are 4-line-to-10-line decoders quite similar to SN54/7442 except for the input code. SN54/7443 accepts the excess-3 BCD code shown in Fig. 8.21. Full decoding of input logic assures that all outputs remain off for any false input conditions.

BCD-to-7-segment Decoder/Drivers. Figures 8.22 and 8.23 show a family of BCD-to-7-segment decoder/drivers: SN54/7446, SN54/7447, SN54/7448, and SN54/7449. All these devices have the same decoding circuitry. The SN54/7446 and SN54/7447 have open-collector output transistors, capable of 20 mA each, which drive lamp segments directly by sinking current to ground. (They differ only in the breakdown ratings of the output transistors.) The other two circuits lack the direct-drive output transistors, being intended for driving lamp segments by sup-

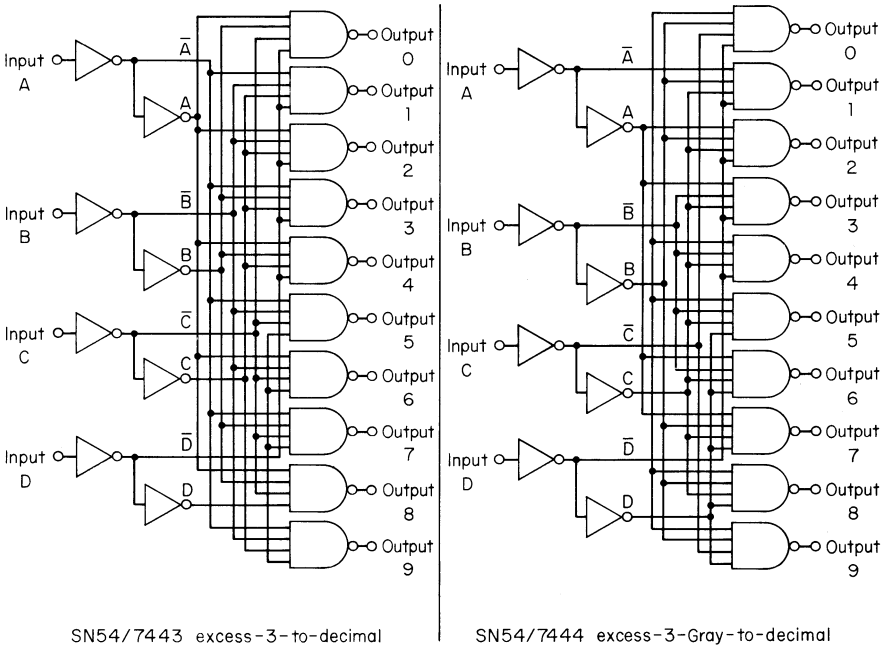

Logic diagram

Truth table

Inputs				Decimal outputs									
D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 8.20. BCD-to-decimal decoder (SN54/7442) and decoder/drivers (SN54/7445 and SN54/74145).

Logic diagram


Input A
Input B
Input C
Input D
$\bar{A}$
A
$\bar{B}$
B
$\bar{C}$
C
$\bar{D}$
D
Output 0
Output 1
Output 2
Output 3
Output 4
Output 5
Output 6
Output 7
Output 8
Output 9

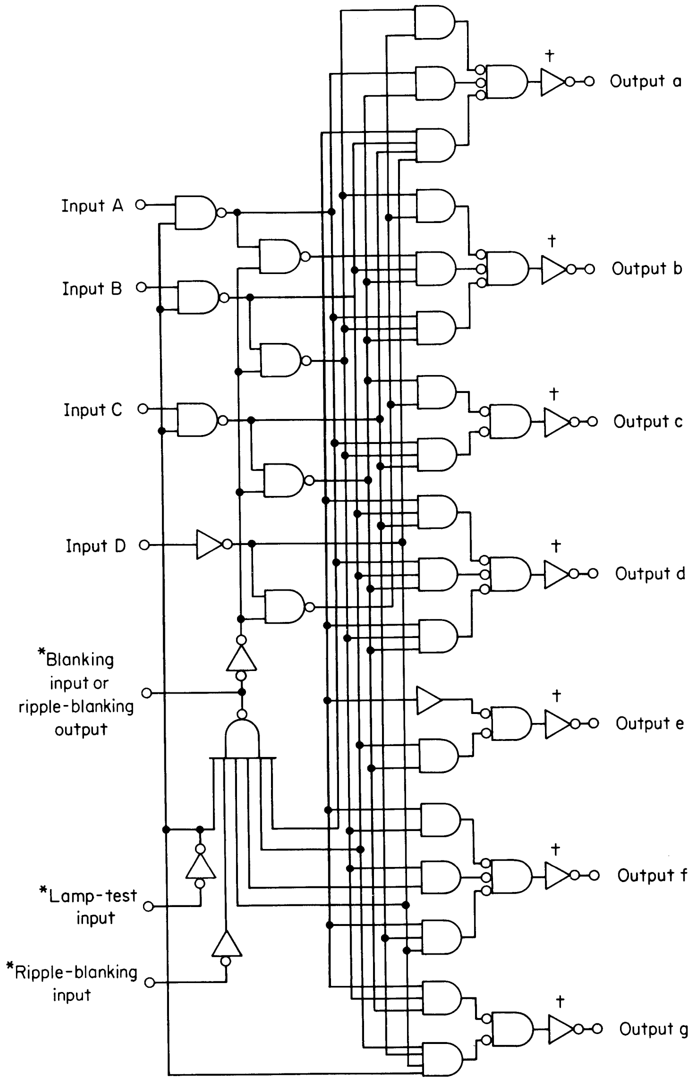
SN54/7443 excess-3-to-decimal

SN54/7444 excess-3-Gray-to-decimal

Truth table

SN54/7443 Excess-3-inputs				SN54/7444 Excess-3-Gray inputs				Decimal outputs									
D	C	B	A	D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	0	1	0	1	1	1	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1
0	1	1	0	0	1	0	1	1	1	1	0	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1
1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1
1	1	0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	0
1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 8.21. Other 4-line-to-10-line decoders: SN54/7443 and SN54/7444.


Input A
Input B
Input C
Input D
*Blanking input or ripple-blanking output
*Lamp-test input
*Ripple-blanking input
†
Output a
Output b
Output c
Output d
Output e
Output f
Output g

* SN54/7449 do not have these features

† SN54/7448 and -49 do not have output inverters

Fig. 8.22. Logic diagram for BCD-to-7-segment decoder/drivers: SN54/7446, SN54/7447, SN54/7448, and SN54/7449.

Inputs								Outputs							Note
Decimal or function	LT*	RBI*	D	C	B	A	BI/RBO*	$a^{\dagger}$	$b^{\dagger}$	$c^{\dagger}$	$d^{\dagger}$	$e^{\dagger}$	$f^{\dagger}$	$g^{\dagger}$	
0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	1	X	0	0	0	1	1	0	1	1	0	0	0	0	1
2	1	X	0	0	1	0	1	1	1	0	1	1	0	1	
3	1	X	0	0	1	1	1	1	1	1	1	0	0	1	
4	1	X	0	1	0	0	1	0	1	1	0	0	1	1	
5	1	X	0	1	0	1	1	1	0	1	1	0	1	1	
6	1	X	0	1	1	0	1	0	0	1	1	1	1	1	
7	1	X	0	1	1	1	1	1	1	1	0	0	0	0	
8	1	X	1	0	0	0	1	1	1	1	1	1	1	1	
9	1	X	1	0	0	1	1	1	1	1	0	0	1	1	
10	1	X	1	0	1	0	1	0	0	0	1	1	0	1	
11	1	X	1	0	1	1	1	0	0	1	1	0	0	1	
12	1	X	1	1	0	0	1	0	1	0	0	0	1	1	
13	1	X	1	1	0	1	1	1	0	0	1	0	1	1	
14	1	X	1	1	1	0	1	0	0	0	1	1	1	1	
15	1	X	1	1	1	1	1	0	0	0	0	0	0	0	
BI	X	X	X	X	X	X	0	0	0	0	0	0	0	0	2
RBI*	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3
LT*	0	X	X	X	X	X	1	1	1	1	1	1	1	1	4

*SN54/7449 do not have these features

†For SN54/7446 and SN54/7447, invert these entries

X = Don't care (input may be high or low)

Notes: 1. BI/RBO is wire-AND logic serving as blanking input (BI) and/or ripple-blanking output (RBO).* The blanking input must be open or held at a logical 1 when output functions 0 through 15 are desired. Ripple-blanking input (RBI)* must be open or at a logical 1 during the decimal 0 input.

2. When a logical 0 is applied to the blanking input (forced condition), all segment outputs go to a logical $0^{\dagger}$ regardless of the state of any other input condition.

3. When ripple-blanking input (RBI)* is at a logical 0 and A = B = C = D = logical $0^{\dagger}$ all segment outputs go to a logical 0 and the ripple-blanking output goes to a logical 0 (response condition).

4. When blanking input/ripple-blanking output is open or held at a logical 1 and a logical 0 is applied to lamp-test input,* all segment outputs go to a logical $1.^{\dagger}$

Fig. 8.23. Truth table for BCD-to-7-segment decoder/drivers: SN54/7446, SN54/7447, SN54/7448, and SN54/7449.

plying logical 1 outputs to intermediate logic circuits or driver transistors. (The SN54/7448 has a resistive-pull-up output; the SN54/7449 has open-collector outputs and lacks the auxiliary inputs for strobing, blanking zeros, and testing the lamp.)

4-line-to-16-line Decoder/Demultiplexer. The SN54/74154 is a monolithic 4-line-to-16 line decoder using TTL circuitry to decode four binary inputs into one of sixteen mutually exclusive outputs when both the data and enable inputs are low. (See Figs. 8.24 and 8.25.) SN74154 may be used also as a demultiplexer by using the four input lines to address the output line, passing data from the data input to the selected output when enable input is low. When enable is high, all outputs are high.

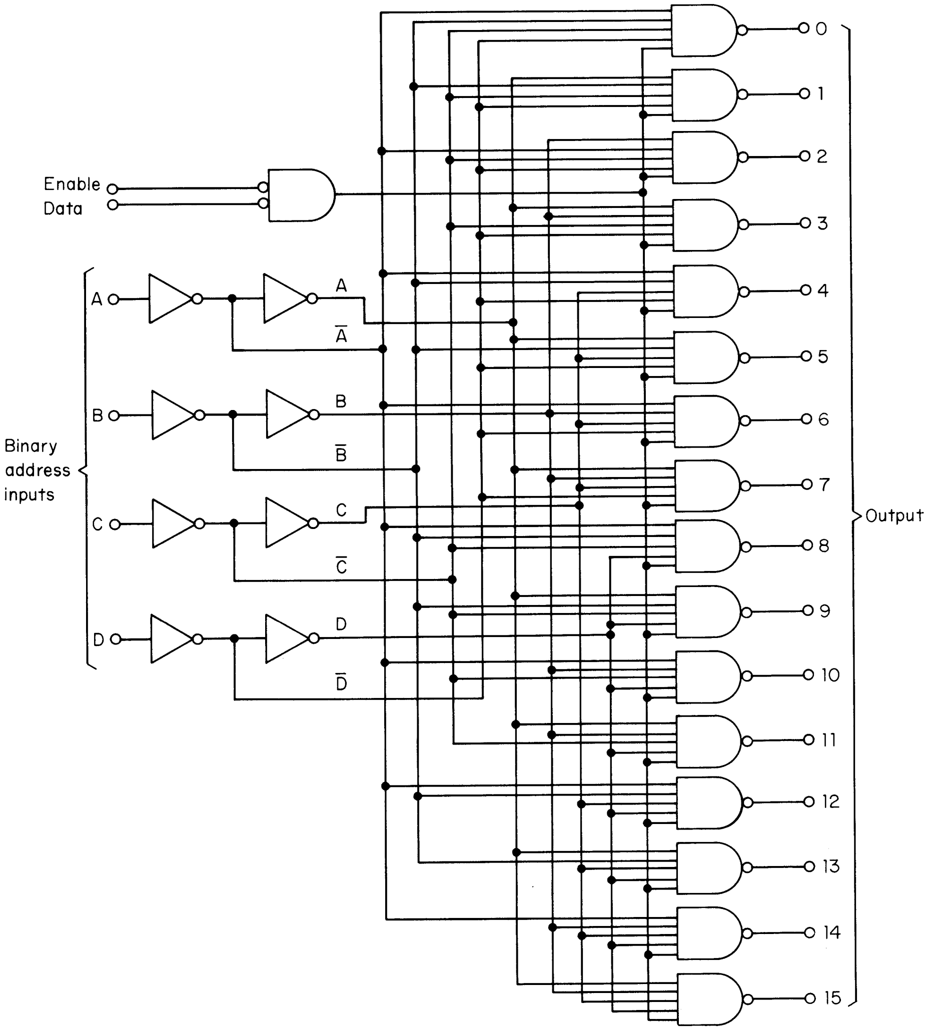

Enable
Data
Binary address inputs
A
B
C
D
A
$\bar{A}$
B
$\bar{B}$
C
$\bar{C}$
D
$\bar{D}$
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Output

Fig. 8.24. Logic diagram for 4-line-to-16-line decoder/demultiplexer, SN54/74155 and SN54/74156.

Inputs						Outputs															
		Address																			
Enable	Data	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 8.25

The SN54/74154 will accept any 4-bit binary code by selecting the correct true output. This fact allows the designer to use the circuit in applications where uncommon 4-bit binary words must be decoded. Also, since all binary input states are explicitly decoded, false data are rejected when fewer than 16 states are used.

2-line-to-4-line or 3-line-to-8-line Decoder/Demultiplexers. SN54/74155 and SN54/74156 can be used as either two 2-line-to-4-line decoders or one 3-line-to-8-line decoder. (See Figs. 8.26 and 8.27.) When both sections are enabled by the strobes, the common binary address inputs sequentially select and route associated input data to the appropriate output of each section. Individual strobes permit activating or inhibiting each of the 4-bit sections as desired. Data applied to input 1*C* are inverted, whereas data applied to 2*C* are not; therefore, an inverter is unnecessary in the 3-line-to-8-line connection.

SN54/74155, with its totem-pole outputs, is rated for a fan-out of 10 TTL loads. The SN74156, with its open-collector outputs, is rated to sink 16 mA at a low-level output of less than 0.4 V.

8.3 APPLICATIONS OF DECODERS

A common application of decimal decoders is in counter systems. A simple system of this sort is shown in Fig. 8.28. Initially the counter is reset to 0 and the hold input is low. When the hold input goes to 1, the counter is enabled and the cycle begins. If the clock repetition rate is 9 MHz and the hold remains at logical 1 for 100 μs, then the number 900 is stored in the counter at the end of a timing cycle. Hold should then remain low for about 1 sec, so that digits may be viewed. A viewing time of 1 sec allows the observer to notice changes in the least significant digits as cycling occurs. Displays used in this example are of the filament-bulb type and cannot usually be driven directly from decoder outputs. Both SN75450 and SN75451 make good display drivers, because they will sink up to 300 mA with a maximum V_{CE} of 30 V when their integral transistors and NAND gates are directly connected as in Fig. 8.28.

The major disadvantage of this simple system is the visual effect with slow clock frequencies. Observers notice display outputs changing as the counter sequences. The outputs can be disabled during a slow counting cycle, but this may cause noticeable blinking of displays. The system shown in Fig. 8.29 eliminates blinking of display outputs by storing information in an SN7474 *D*-type flip-flop. (An SN7475 4-bit latch also can be used in the place of two SN7474s.) Counting information is displayed continuously, changing only upon the strobe pulse, when

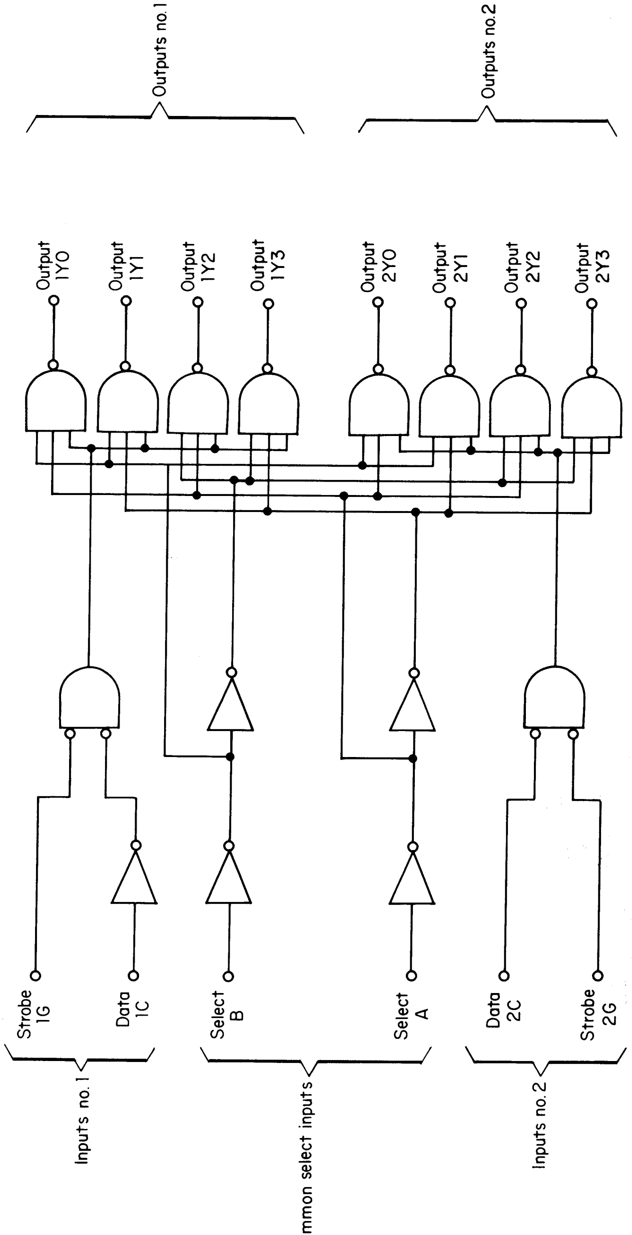
2-line-to-4-line decoders, or 1-line-to-4-line demultiplexers

Inputs						Outputs							
Select		Strobe		Data									
A	B	1G	2G	1C	2C	1Y0	1Y1	1Y2	1Y3	2Y0	2Y1	2Y2	2Y3
X	X	1	1	X	X	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	1	1	1	0	1	1	1
0	0	0	0	1	1	0	1	1	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	0	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	1	1	0	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
1	1	0	0	0	0	1	1	1	1	1	1	1	0
1	1	0	0	1	1	1	1	1	0	1	1	1	1
X	X	X	X	0	1	1	1	1	1	1	1	1	1

3-line-to-8-line decoders, or 1-line-to-8-line demultiplexers

Inputs						Outputs							
Address				Input or Strobe									
Select		Data											
A	B	1C	2C	1G	2G	2Y0	2Y1	2Y2	2Y3	1Y0	1Y1	1Y2	1Y3
X	X	X	X	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	1	0	1	1	1	1	1	1
0	1	0	0	0	0	1	1	0	1	1	1	1	1
1	1	0	0	0	0	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1	0	1	1
0	1	1	1	0	0	1	1	1	1	1	1	0	1
1	1	1	1	0	0	1	1	1	1	1	1	1	0

Fig. 8.26. Truth tables for Fig. 8.27.


Inputs no. 1
Strobe 1G
Data 1C
Common select inputs
Select B
Select A
Inputs no. 2
Data 2C
Strobe 2G
Output 1Y0
Output 1Y1
Output 1Y2
Output 1Y3
Output 2Y0
Output 2Y1
Output 2Y2
Output 2Y3
Outputs no. 1
Outputs no. 2

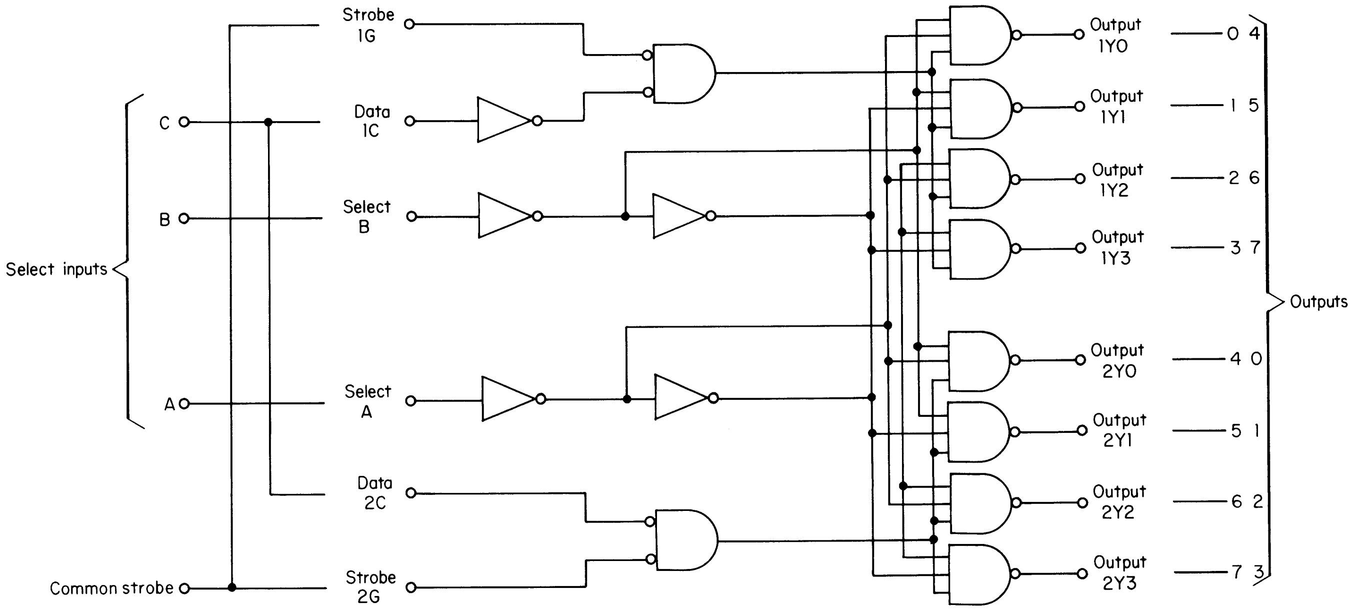

Strobe 1G
Data 1C
Select B
Select A
Data 2C
Strobe 2G
C
B
A
Select inputs
Common strobe
Output 1Y0
Output 1Y1
Output 1Y2
Output 1Y3
Output 2Y0
Output 2Y1
Output 2Y2
Output 2Y3
0 4
1 5
2 6
3 7
4 0
5 1
6 2
7 3
Outputs

Figure 8.27

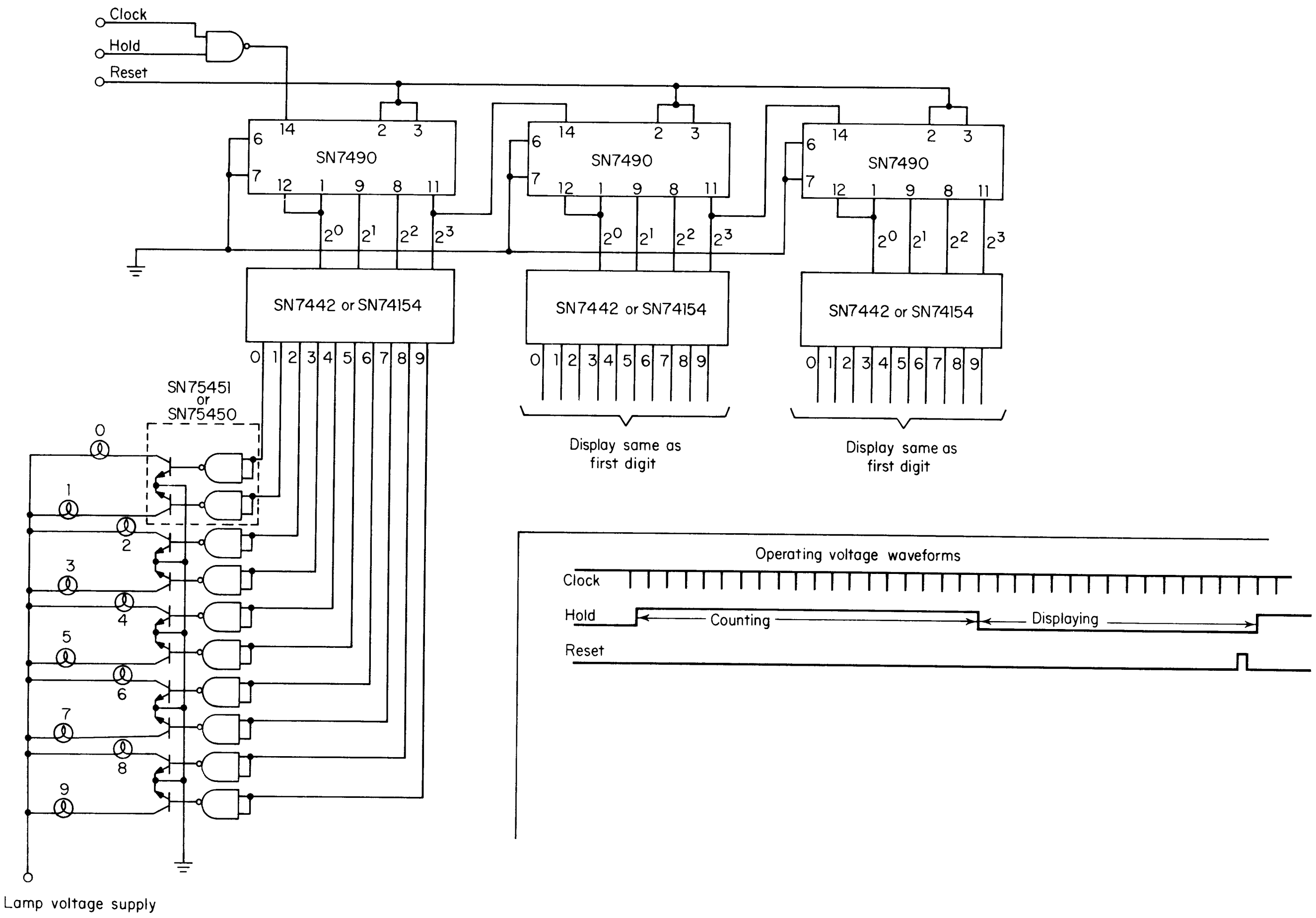

Clock
Hold
Reset
SN7490
6
14
2
3
7
12
1
9
8
11
2^0
2^1
2^2
2^3
SN7442 or SN74154
0 1 2 3 4 5 6 7 8 9
SN75451
or
SN75450
Display same as
first digit
Lamp voltage supply
Operating voltage waveforms
Clock
Hold
Counting
Displaying
Reset

Fig. 8.28. Decoders used in a clock-pulse counter and display system.

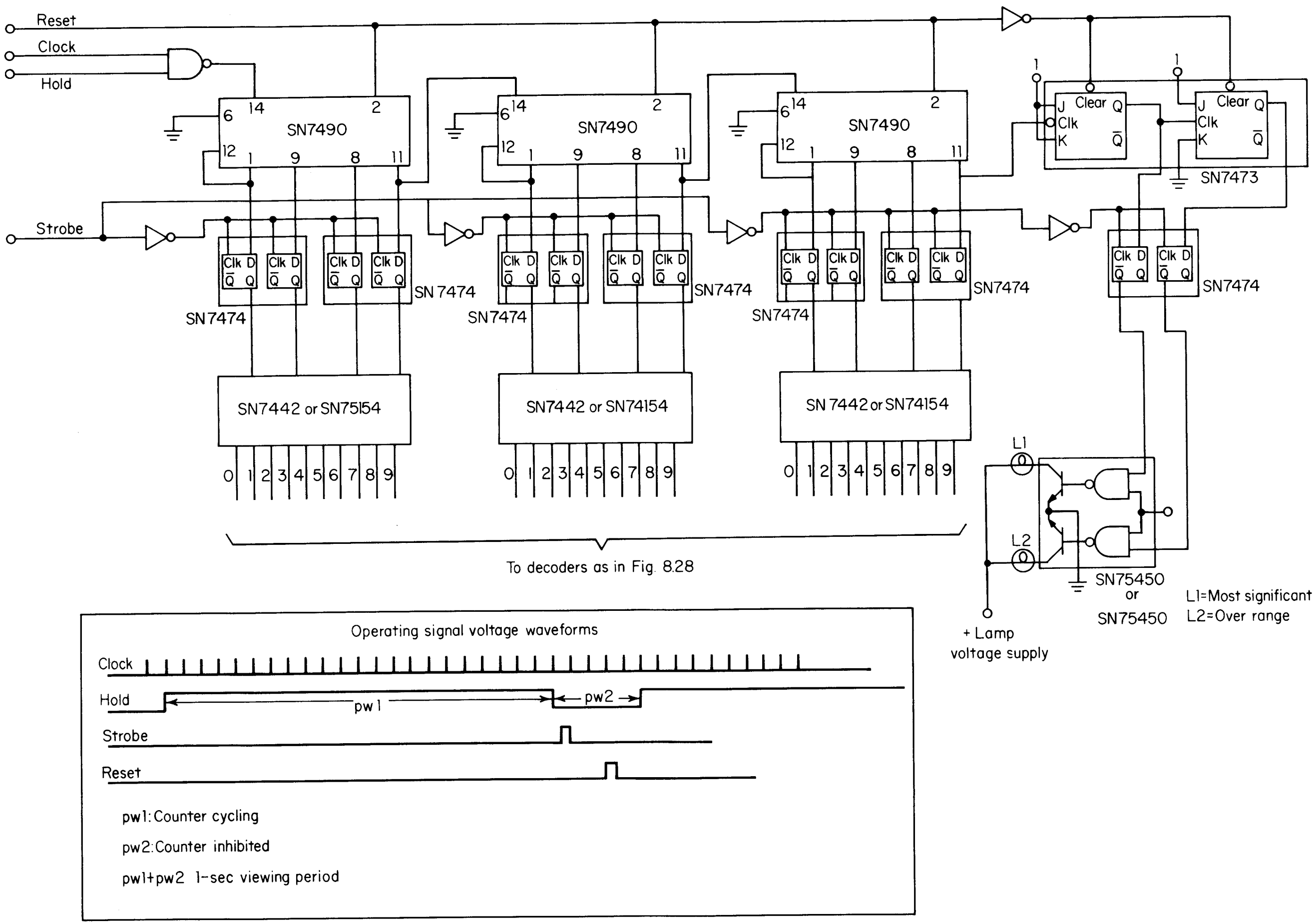

Reset
Clock
Hold
Strobe
6
14
2
12
1
9
8
11
SN7490
Clk D
Q̄ Q
SN7474
SN7442 or SN75154
SN7442 or SN74154
SN 7442 or SN74154
0 1 2 3 4 5 6 7 8 9
To decoders as in Fig. 8.28
J Clear Q
Clk
K Q̄
SN7473
1
L1
L2
SN75450
or
SN75450
L1=Most significant
L2=Over range
+ Lamp
voltage supply
Operating signal voltage waveforms
Clock
Hold
pw 1
pw2
Strobe
Reset
pw1: Counter cycling
pw2: Counter inhibited
pw1+pw2 1-sec viewing period

Fig. 8.29. Improved clock-pulse counter and display system to use decoders.

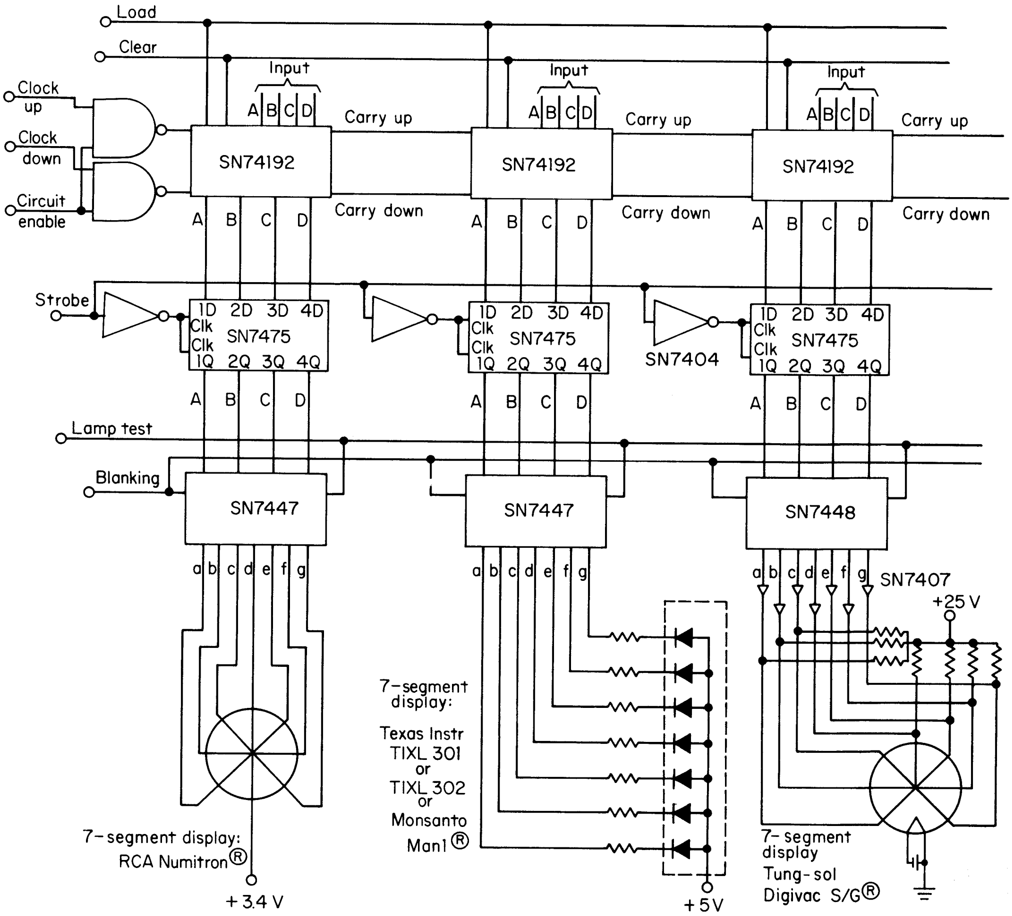

Load
Clear
Clock up
Clock down
Circuit enable
Input
A B C D
Carry up
SN74192
A B C D
Carry down
Strobe
1D 2D 3D 4D
Clk
Clk
SN7475
1Q 2Q 3Q 4Q
SN7404
Lamp test
Blanking
SN7447
SN7448
a b c d e f g
SN7407
+25 V
7-segment display: RCA Numitron®
+3.4 V
7-segment display: Texas Instr TIXL 301 or TIXL 302 or Monsanto Man1®
+5V
7-segment display Tung-sol Digivac S/G®

Fig. 8.30. Decoders used in a synchronous up/down counter with 7-segment display system.

the timing cycle has ended and correct information is in the counter. The strobe pulse should not occur more than once per second so that digit changes are easily observed. Lamp *L*1 displays a most significant 1, and lamp *L*2 indicates when the counter range is exceeded.

Figure 8.30 shows a synchronous counter system using the SN74192 synchronous BCD up/down counter. In this example the 7-segment display devices can be driven directly from decoder outputs. Blanking input can be used to modulate outputs, by rapid interruption, thus allowing intensity control of the displays without changing driving voltage. Lamp test allows verification that all display segments are in working order. When the lamp test is a logical 0, all segments are on.

Another application for decoders is in analog-to-digital converters, such as shown in Fig. 8.31. An analog input voltage $V_{unknown}$ is placed on one side of the SN72709 operational amplifier, and the counter is reset to 0. The op amp output is interrogated to find out whether $V_{unknown} > V_{known}$. If it is, counting continues. Strobe

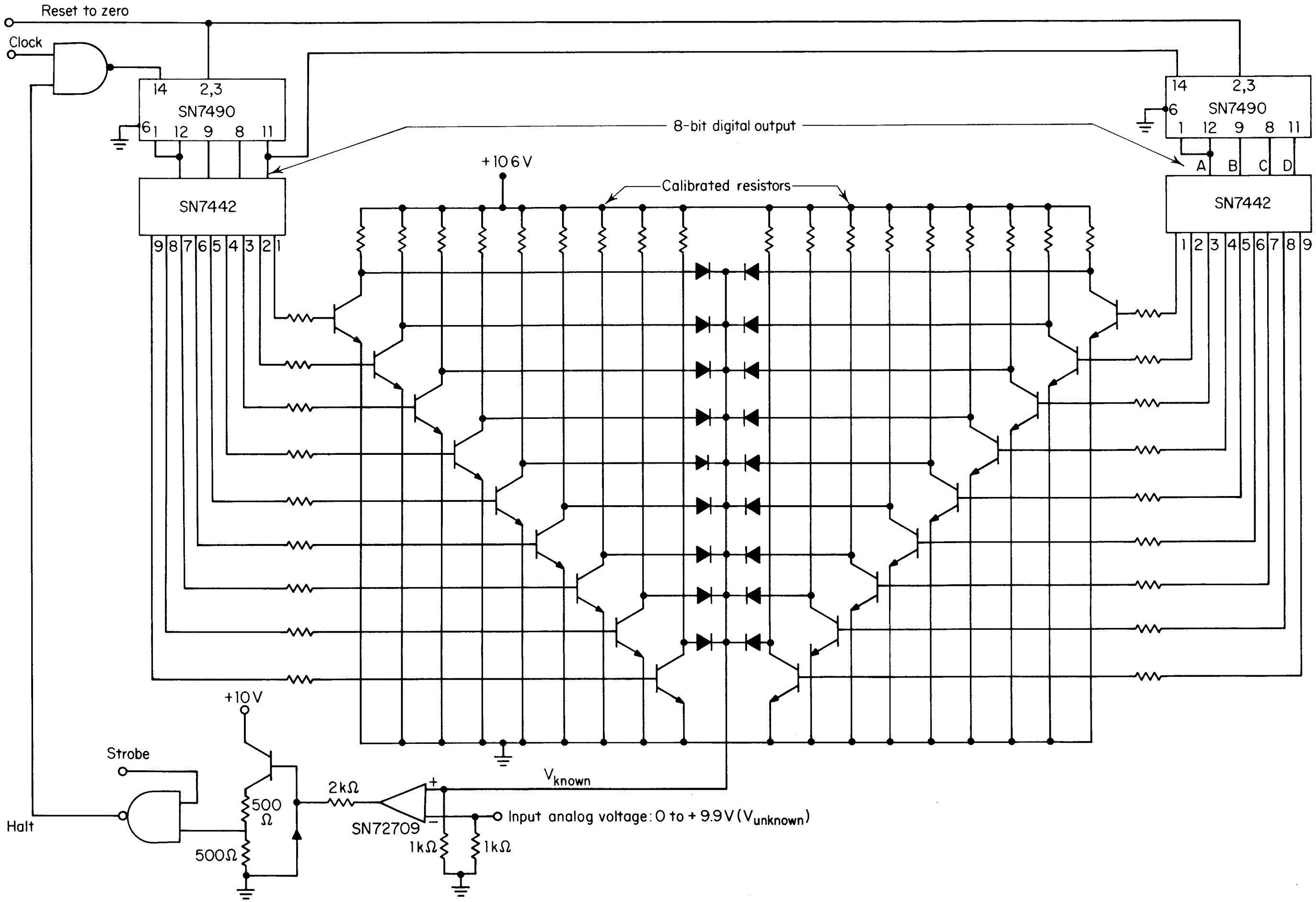

Reset to zero
Clock
14
2,3
SN7490
6
1
12
9
8
11
SN7442
9
8
7
6
5
4
3
2
1
8-bit digital output
+106V
Calibrated resistors
A
B
C
D
+10V
Strobe
Halt
500 Ω
500Ω
2kΩ
SN72709
+
−
1kΩ
1kΩ
V_known
Input analog voltage: 0 to + 9.9V (V_unknown)

Fig. 8.31. Decoders used in an analog-to-digital converter.

input allows interrogation of the op amp output only after it has stabilized. V_{known} is generated in the calibrated resistor matrix, with currents summed at the op amp input. When $V_{unknown} < V_{known}$, the halt output goes to a logical 0, disabling the counter input. If the summing resistors have been chosen properly, the number stored in the counter will approximate $V_{unknown}$. Decoder outputs can then be used to drive a display system.

9

Arithmetic Elements

The essential function of most computers is to perform purely arithmetic operations, and almost all computers employ at least some arithmetic elements. This chapter discusses those TTL integrated-circuit logic elements that are especially applicable to digital arithmetic operations.

9.1 ADDITION OF BINARY NUMBERS

In the binary number system, a quantity is represented by

$$A_n \cdot 2^n + A_{n-1} \cdot 2^{n-1} + A_{n-2} \cdot 2^{n-2} + \cdots + A_2 \cdot 2^2 + A_1 \cdot 2^1 + A_0 \cdot 2^0 + A_{-1} \cdot 2^{-1} + A_{-2} \cdot 2^{-2} \cdots$$

The actual number as written consists of the characteristics only and would be written $A_nA_{n-1}A_{n-2} \cdots A_2A_1A_0$, where each A would have a value of either 1 or 0. To add, it is necessary to arrange two such numbers so that digits with the same weight are compared.

The following shows all combinations for a 1-bit addend and augend:

$$\begin{array}{rrrrll}
0 & 0 & 1 & 1 & \leftarrow \text{Addend} & (A) \\
+0 & +1 & +0 & +1 & \leftarrow \text{Augend} & (B) \\
\hline
0 & 1 & 1 & 10 & \leftarrow \text{Sum} & (S) \\
 & & & \nwarrow & \text{Carry} &
\end{array}$$

Logic equations and a truth table representing these operations are shown in Fig. 9.1.

$$S(\text{sum}) = A\bar{B} + \bar{A}B \qquad (1)$$

$$C(\text{carry}) = AB \qquad (2)$$

The logic equation for the sum is known also as the exclusive-OR function and can be represented also in *Boolean ring algebra* as

$$S = \bar{A}B + A\bar{B} = A \oplus B \qquad (3)$$

Such an adder can be implemented by the elementary circuit of Fig. 9.2.

Addend A	Augend B	Sum S	Carry C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = A\bar{B} + \bar{A}B = A \oplus B$

$C = AB$

Figure 9.1

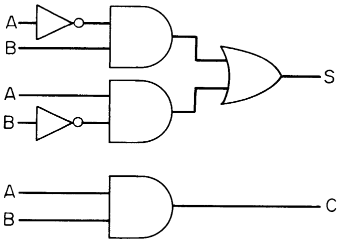


Fig. 9.2. Half-adder logic diagram.

In the more general case of addition of multibit numbers, it is necessary to provide for a carry input produced by a previous cycle in a serial adder or a previous stage in a parallel adder. Figure 9.3 presents logic equations and a truth table for an adder that provides for a carry input.

To derive the logic equations, the truth table results are plotted on Karnaugh maps (Fig. 9.4).

$$S = ABC_{n-1} + A\overline{BC}_{n-1} + \overline{AB}C_{n-1} + \bar{A}B\bar{C}_{n-1} \tag{4}$$

$$C_n = BC_{n-1} + AC_{n-1} + AB \tag{5}$$

The previous adder with no provision for carry-in is known as a half-adder. The full-adder can be implemented using basic logic elements as shown in Fig. 9.5.

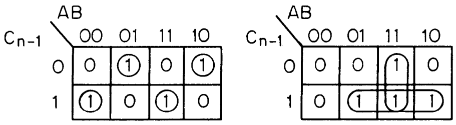
The function of the circuit in Fig. 9.5 is implemented by the SN54/7480, SN54/7482, and SN54/7483 series of complex function adders.

Inputs			Outputs	
Carry-in C_{n-1}	Addend A	Augend B	Sum S	Carry out C_n
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$S = ABC_{n-1} + A\overline{BC}_{n-1} + \overline{AB}C_{n-1} + \bar{A}B\bar{C}_{n-1}$

$C_n = BC_{n-1} + AC_{n-1} + AB$

Figure 9.3



C_{n-1} \ AB	00	01	11	10
0	0	①	0	①
1	①	0	①	0

S = sum

C_{n-1} \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

C_n = carry out

Figure 9.4

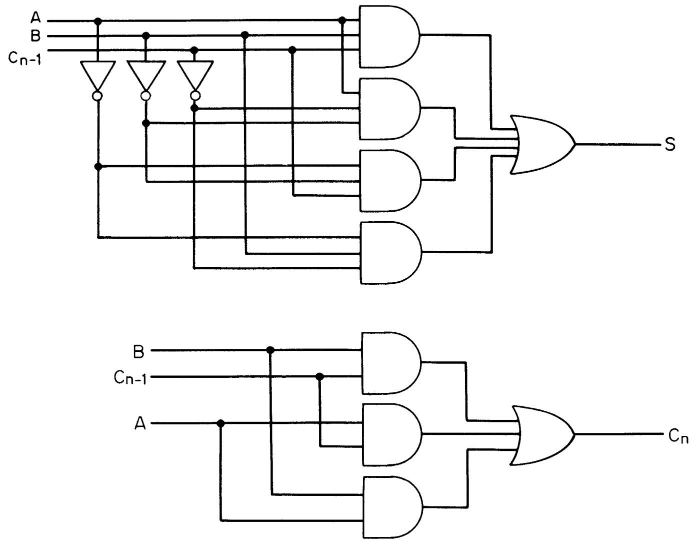

A
B
Cn-1
S
B
Cn-1
A
Cn

Fig. 9.5. Full-adder logic diagram.

9.2 PARALLEL BINARY ADDITION

A number of single-bit full-adders can be arranged as shown in Fig. 9.6 to form a multibit adder. In this circuit arrangement, the carry must be allowed time to propagate through each of the adders before the addition can be considered complete. Although this arrangement can be carried to any length, the carry propagation time makes it unacceptable in many applications.

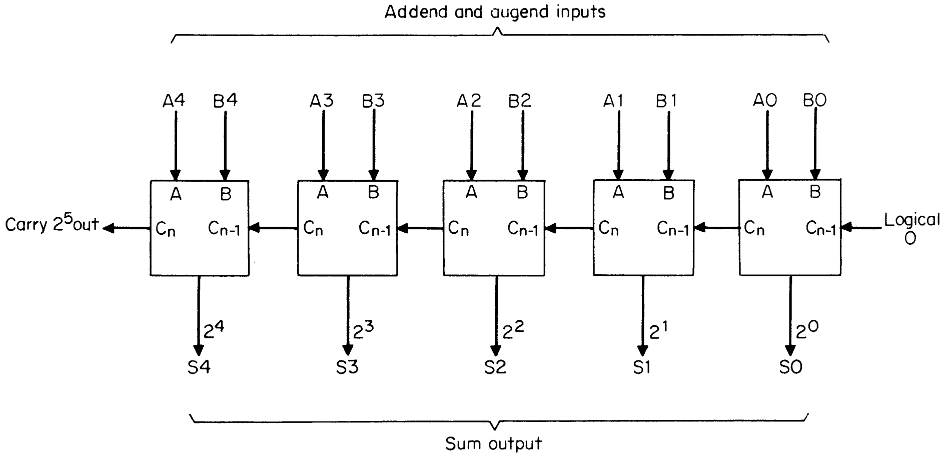

Addend and augend inputs
A4
B4
A3
B3
A2
B2
A1
B1
A0
B0
A
B
Cn
Cn-1
Carry 2^5 out
Logical 0
2^4
2^3
2^2
2^1
2^0
S4
S3
S2
S1
S0
Sum output

Fig. 9.6. Parallel adder, 5 bits.

9.3 SERIAL BINARY ADDER

A serial adder uses one full-adder and performs addition of only one bit position per cycle. The *D*-type flip-flop is required for storage of a possible carry to the next cycle.

There are several options for the shift registers, such as parallel or serial data entry and various connections for input and output. The sum register can be combined with either the addend or the augend register so that the sum displaces the original number at completion. It may also be desirable, where the same number is repeatedly added, to recirculate a number within a register by connecting the output to the input as shown by the dashed line at the addend register in Fig. 9.7.

9.4 SERIES 54/74 TTL ARITHMETIC ELEMENTS

There are presently six complex-function adder packages in the Series SN54/74 logic for 1, 2, and 4 bits.

SN54/7480. The SN54/7480 is a single-bit gated full-adder with gated complementary inputs and complementary sum (Σ and $\overline{\Sigma}$) outputs. The inverted carry output is designed for medium- and high-speed multiple-bit parallel add/serial carry applications. The circuit utilizes diode transistor logic (DTL) for the gated inputs, and high-speed, high-fan-out TTL for the sum and carry outputs. The circuit is compatible with both DTL and TTL logic. The implementation of a single-inversion, high-speed, Darlington-connected, serial carry circuit minimizes the need for extensive look-ahead and carry-cascading circuits. The equivalent logic diagram is shown in Fig. 9.8.

The SN7480 is designed specifically for multibit addition or subtraction operations without external gates or inverters. In the SN7480, two methods are used to reduce

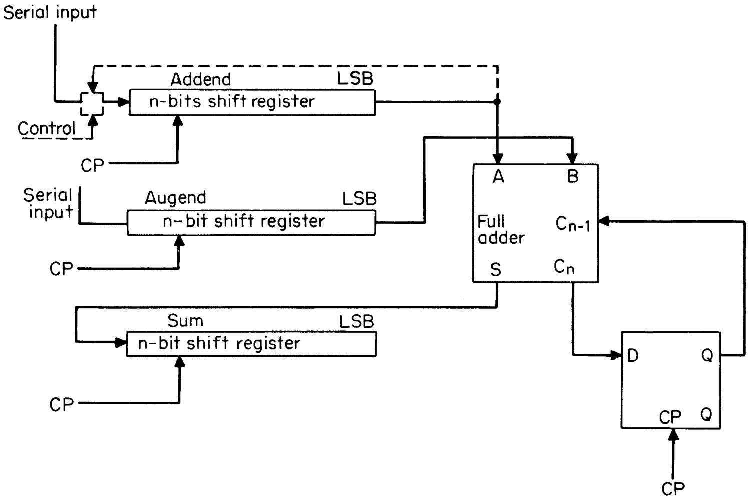


Fig. 9.7. Serial adder.

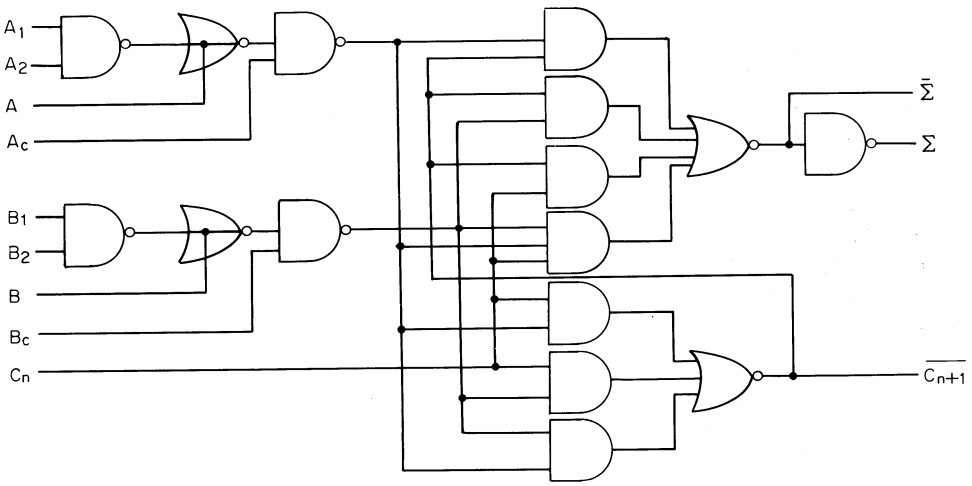
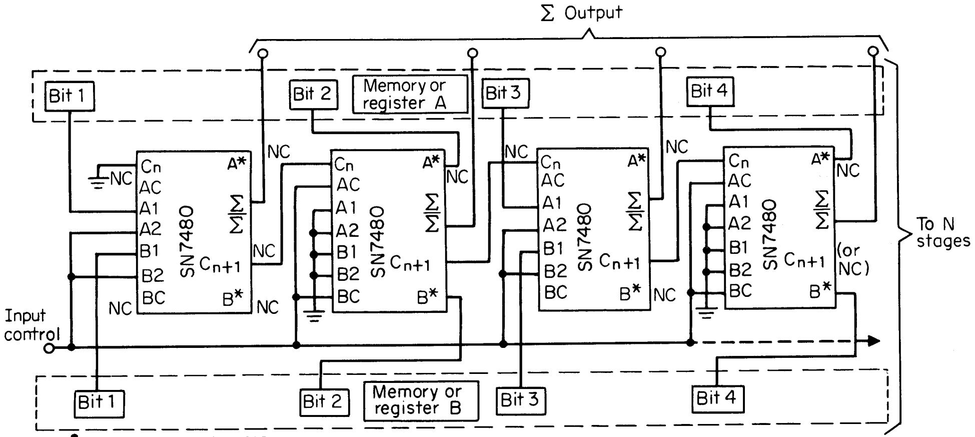

A1
A2
A
Ac
B1
B2
B
Bc
Cn
$\overline{\Sigma}$
Σ
$\overline{C_{n+1}}$

Fig. 9.8. SN7480 gated full-adder.

the carry delay. The carry circuit employs a high-speed Darlington output, and the logic gating has only one inversion between the carry input C_n and the carry output C_{n+1}. This logic configuration results in an inverted carry output. To counteract this inverted input to the following stage without sacrificing propagation time for the carry, gates are provided within the circuit to invert A and B inputs and the sum output. This interconnection method is illustrated by bit 2 and bit 4 of the adder in Fig. 9.9. The inverted carry output is a true carry from bit 2 and bit 4, permitting the use of noninverted A and B inputs for the odd-numbered bits.


Σ Output
Bit 1
Bit 2
Memory or register A
Bit 3
Bit 4
Cn
AC
A1
A2
B1
B2
BC
A*
Σ
$\overline{\Sigma}$
Cn+1
B*
SN7480
NC
Input control
To N stages
(or NC)
Bit 1
Bit 2
Memory or register B
Bit 3
Bit 4

Note: Functions noted as NC are open

Fig. 9.9. N-bit binary adder using the SN7480.

The input control is used to disable the A and B inputs when memory or register information is being shifted. A logical 0 applied to this line will bring each sum to a 0 condition and maintain this level regardless of the state of the input information into each bit. Input control is applied to $A2$ and $B2$ of odd-numbered bits and to A_C and B_C of even-numbered bits. These alternating patterns are necessary to coincide with the varying input sequence they control. The A^* and B^* inputs require the use of a gate with open-collector output such as the SN7401, SN7403, and SN74H01 as a signal source.

SN54/7482 and SN54/7483. The SN7482 (Fig. 9.10) and SN7483 (Fig. 9.11) are respectively 2- and 4-bit binary adders that perform parallel addition with internally connected ripple-through (serial) carry. The sum outputs are provided for each bit, and the resultant carry-out $C2$ or $C4$ is obtained from the last full-adder stage. The basic logic configuration for the sum and carry is the same as for the SN7480, but there is no gating for inputs on the SN7482 and SN7483 adders. The complement of the carry-out is used for the carry input of higher-order bits. The consequence is that for even-numbered bits, it is required to invert the inputs, and for even-numbered sum outputs, an inverter is (unlike odd-numbered sums) not required.

In addition to the adders, there are other units which are quite often associated with digital arithmetic functions. The SN7486 quadruple 2-input exclusive-OR and the SN74H87 4-bit true/complement gate are such elements.

SN54/7486. The 54/7486 package contains four 2-input exclusive-OR gates.

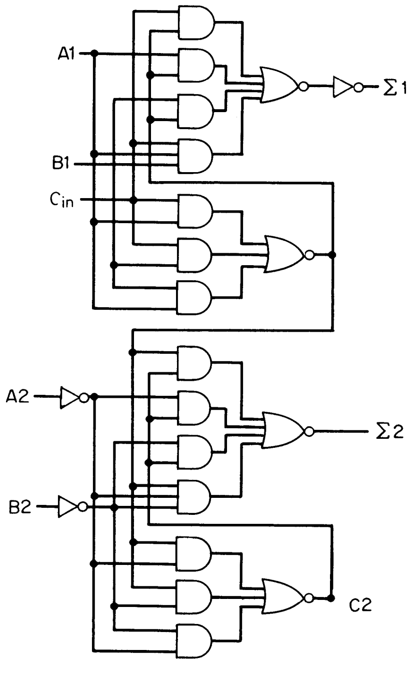


Fig. 9.10. SN54/7482 2-bit adder, logic diagram.

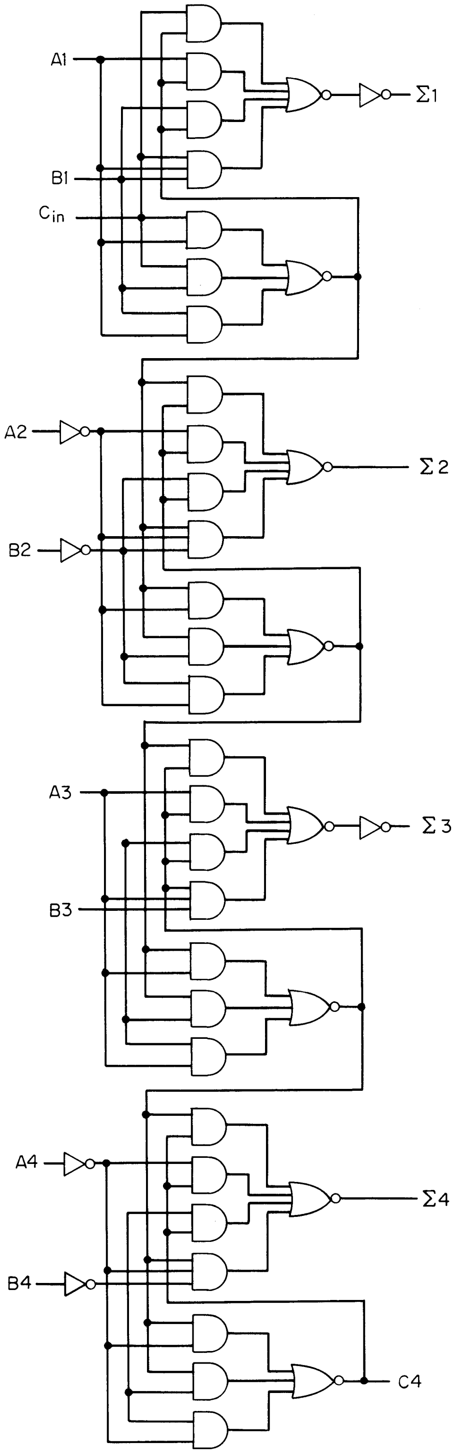
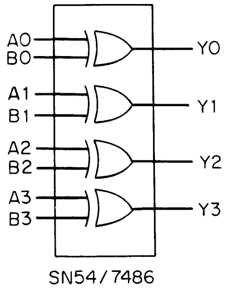

A1
B1
C_in
Σ1
A2
B2
Σ2
A3
B3
Σ3
A4
B4
Σ4
C4

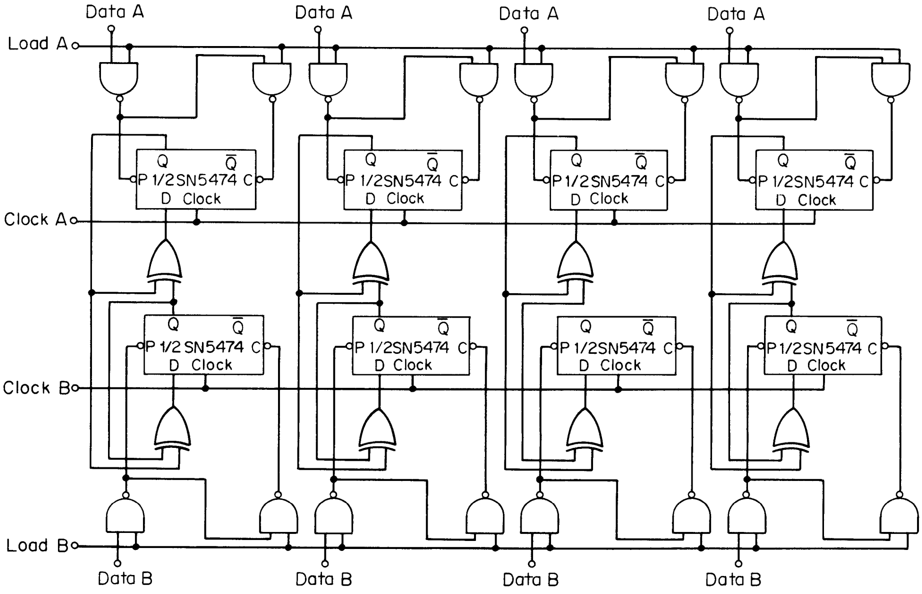
Fig. 9.11. SN54/7483 4-bit adder, logic diagram.


A0
B0
Y0
A1
B1
Y1
A2
B2
Y2
A3
B3
Y3
SN54/7486

Truth table
SN54/7486

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Fig. 9.12. SN54/7486 quad exclusive-OR.


Data A
Data A
Data A
Data A
Load A
Q
$\overline{Q}$
P 1/2SN5474 C
D Clock
Clock A
Clock B
Load B
Data B
Data B
Data B
Data B

The following sequence will exchange the contents

Sequence	Contents of register A	Contents of register B
Start	A	B
Clock register A	$A \oplus B$	B
Clock register B	$A \oplus B$	$(A \oplus B) \oplus B = A$
Clock register A	$(A \oplus B) \oplus A = B$	A

For a detailed example

Sequence	Contents of register A	Contents of register B
Start	1011 = A	0010 = B
Clock register A	1001	0010
Clock register B	1001	1011 = A
Clock register A	0010 = B	1011 = A

Fig. 9.13. Register exchange using SN54/7486.

The exclusive-OR function of two variables is

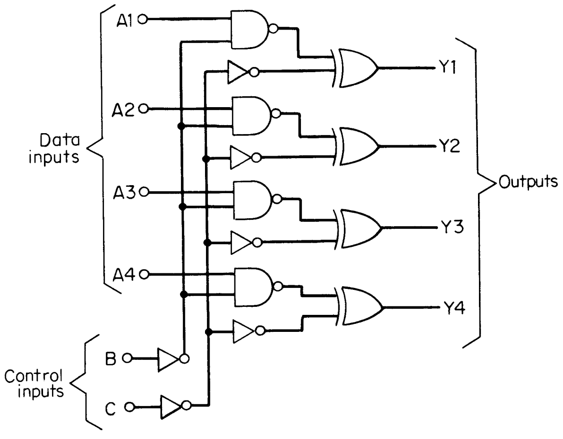
$$Y = A\overline{B} + \overline{A}B \quad \text{or} \quad A \oplus B$$

Refer to Fig. 9.12 for diagram and truth table.

Input clamping diodes and full on-chip buffering are provided to improve circuit performance and simplify system design. The exclusive-OR function is widely used in digital systems, especially in applications requiring error detection, comparison schemes, and counting.

Figure 9.13 shows an application of the SN7486 in a register exchange function. The contents of register *A* and register *B* are exchanged by using the exclusive-OR with the SN5474 duel *D*-type flip-flops. The data are loaded into the register from the asynchronous inputs on the SN5474. The exclusive-OR can be used also as a controllable true/complement unit as shown in Fig. 9.20.

SN54/74H87. The SN54/74H87 performs the true/complement and 0/1 functions, and the SN74H87 with two control lines transfers 4-bit input data (or their complement) to the output. The control inputs can also set all outputs to 0 or 1. Input clamping diodes and control-line input buffers are provided to improve circuit performance and simplify system design. In addition to the usual application shown in Fig. 9.18, the SN74H87 has several other uses arising from its ability to produce all 1s or 0s. Since all 1s is the 2's complement of a binary −1, a binary number can be decreased by 1 by adding all 1s to the number. When it is desirable to perform no addition, the SN74H87 can be set to all 0s. Figure 9.14 is the truth table and logic diagram for the true/complement gate.



Truth table SN74H87

Control inputs		Output			
B	C	Y1	Y2	Y3	Y4
0	0	$\overline{A1}$	$\overline{A2}$	$\overline{A3}$	$\overline{A4}$
0	1	A1	A2	A3	A4
1	0	1	1	1	1
1	1	0	0	0	0

Fig. 9.14. SN54/74H87 4-bit true/complement 1/0 gate.

9.5 BINARY REPRESENTATIONS FOR COMPUTER ARITHMETIC

This section shows the development of algorithms required for performing binary addition and subtraction with the three principal notations used in binary arithmetic. These three notations are the sign and magnitude, 1's complement, and 2's complement. The 2's complement is known as a radix complement for binary numbers, just as a 10's complement is a radix complement in the decimal system. The radix complement of a number is defined by Eq. (7).

$$A + A^* = r^N \tag{6}$$

or

$$A^* = r^N - A \tag{7}$$

where A = number for which the complement is to be found
A^* = radix complement of A
r = radix ($r = 10$ for decimal and 2 for binary)
N = number of digits in A

To find the 10's complement of 2,245,

$$\begin{array}{rr}
N = & 4 \\
A^* = & 10^4 - 2{,}245 \\
 & 10{,}000 \\
 & \underline{-2{,}245} \\
A^* = & 7{,}755
\end{array}$$

Using binary numbers, find the 2's complement of 10110:

$$\begin{array}{rrl}
N = & 5 & \\
A^* = & 2^5 - 10110 & \\
 & 100000 & r^N \\
 & \underline{-10110} & A \\
A^* = & 01010 &
\end{array}$$

The 1's complement for binary numbers and the 9's complement for decimal numbers are known as diminished radix complements; they are defined by Eq. (9).

$$B + B^* = r^N - 1 \tag{8}$$

or

$$B^* = r^N - 1 - B \tag{9}$$

where B = number for which the complement is to be found
B^* = diminished radix complement of B

For an example using binary numbers, determine the 1's complement of 10110:

$$\begin{array}{rrl}
N = & 5 & \\
B^* = & 2^5 - 1 - 10110 & \\
 & 11111 & r^N - 1 \\
 & \underline{-10110} & B \\
B^* = & 01001 &
\end{array}$$

Notice that this operation is equivalent to interchanging all 0s with 1s and all 1s with 0s.

Binary point notation will be adopted in the following discussions; this means that all numbers are considered less than 1, and that to the left of the binary point is the sign-bit. A negative number has a 1 in the sign position and a positive number has a 0 in the sign position. This convention is adopted here because of its advantages in binary arithmetic and because it is used in many modern computers.

Sign and Magnitude Notation. In sign and magnitude notation, numbers of the same magnitude are identical to the right of the binary point; they differ only in the sign-bit. Thus,

$$+26(2^{-5}) = 0.11010 \tag{10}$$

$$-26(2^{-5}) = 1.11010 \tag{11}$$

1's Complement Notation. A positive number is the same in all of the three systems discussed.

Negative and positive numbers of equal magnitude in 1's complement are exact complements of each other. Thus,

$$+26(2^{-5}) = 0.11010 \tag{12}$$

$$-26(2^{-5}) = 1.00101 \tag{13}$$

2's Complement Notation. A negative number is formed by taking its positive counterpart and subtracting it from 2. We then have

$$+26(2^{-5}) = 0.11010 \tag{14}$$

$$-26(2^{-5}) = 1.00110 \tag{15}$$

A simple rule for finding the 2's complement is to find the 1's complement and add a 1 to the LSB (least significant bit). Another method which can be of advantage is to start at the right and examine bits of the number in sequence. For each 0 in the number, place a 0 in the complement being generated. When the first 1 is reached, place a 1 in the complement. Thereafter, for each 0 place a 1 in the complement, and for each 1 in the original number, place a 0 in the complement for all bits including the sign-bit.

Sign and Magnitude Addition and Subtraction. The addition of positive numbers is the same in all three notations. One precaution to be observed is that the sum must not exceed 1, or the number will exceed the word size of the register.

In Fig. 9.15, for the sign and magnitude representation, note that the (2^{-5}) has been dropped for all the binary numbers; this convention will be used throughout. Subtraction is performed in a parallel binary machine by complement addition for all representations. This can be seen in Figs. 9.15, 9.16, and 9.17, where one number is actually subtracted from another. For sign and magnitude subtraction, the usual procedure is to change the sign of the subtrahend and perform the appropriate addition as shown in Fig. 9.15. For example, assume it is desired to subtract a positive number (subtrahend) from a positive number (minuend). It is necessary to change the sign of the subtrahend, that is, complement its sign-bit only, and then proceed according to the rules of addition, substituting the terms minuend, subtrahend, and remainder for augend, addend, and sum, respectively.

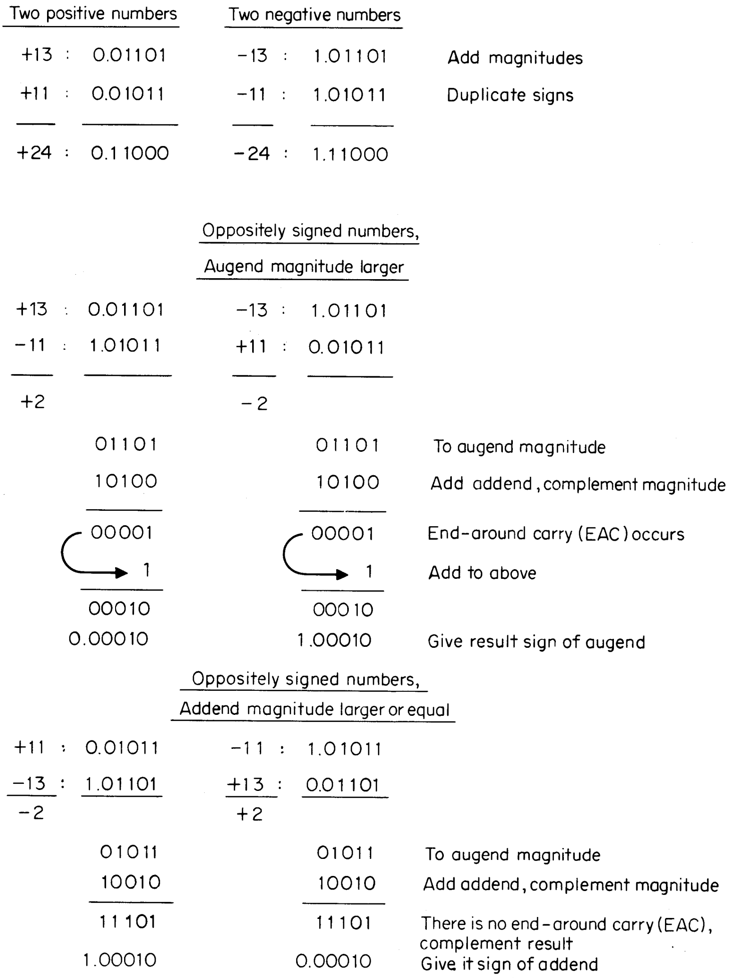

Two positive numbers
Two negative numbers
+13 : 0.01101
-13 : 1.01101
Add magnitudes
+11 : 0.01011
-11 : 1.01011
Duplicate signs
+24 : 0.11000
-24 : 1.11000
Oppositely signed numbers,
Augend magnitude larger
+13 : 0.01101
-11 : 1.01011
-13 : 1.01101
+11 : 0.01011
+2
-2
01101
10100
01101
10100
To augend magnitude
Add addend, complement magnitude
00001
1
00001
1
End-around carry (EAC) occurs
Add to above
00010
0.00010
00010
1.00010
Give result sign of augend
Oppositely signed numbers,
Addend magnitude larger or equal
+11 : 0.01011
-13 : 1.01101
-11 : 1.01011
+13 : 0.01101
-2
+2
01011
10010
01011
10010
To augend magnitude
Add addend, complement magnitude
11101
11101
There is no end-around carry (EAC), complement result
1.00010
0.00010
Give it sign of addend

Fig. 9.15. Sign and magnitude addition.

1's Complement Addition and Subtraction. Two positive numbers are added in the same manner in all representations. For negative numbers, each possibility is shown in Fig. 9.16; note that the sign-bit is also added. For subtraction, the 1's complement of the subtrahend (including the sign-bit) is added to the minuend; and the EAC (end-around-carry) operation is performed. This actually is equivalent to using the old rule that says "change the sign of the subtrahend and add." The addition is then just as shown in Fig. 9.16.

2's Complement Addition and Subtraction. Addition of positive numbers is the same for all notations. For 2's complement addition the numbers (including the sign-bit) are added, and a carry-out from the sign-bit is always ignored. Subtraction, as illustrated in Fig. 9.17, is performed by adding the 1's complement of the subtrahend to the minuend, and adding 1 in the LSB position. This again is

Two positive numbers

$$\begin{array}{rl} +13: & 0.01101 \\ +11: & 0.01011 \\ \hline +24: & 0.11000 \end{array}$$

← Add numbers including sign bit →
retain positive sign

Two negative numbers

$$\begin{array}{rll} -13: & 1.10010 & \\ -11: & 1.10100 & \\ \hline & 1.00110 & \\ & \hookrightarrow\ 1 & \text{Add in EAC} \\ \hline -24: & 1.00111 & \end{array}$$

Oppositely signed numbers,
negative result

$$\begin{array}{rlrll} -13: & 1.10010 & +11: & 0.01011 & \text{Add numbers including sign bit} \\ +11: & 0.01011 & -13: & 1.10010 & \\ \hline -2 & 1.11101 & -2 & 1.11101 & \text{No EAC} \\ & & & & \text{Result is correct} \end{array}$$

Oppositely signed numbers,
positive result

$$\begin{array}{rlrll} +13: & 0.01101 & -11: & 1.10100 & \text{Add numbers including sign bit} \\ -11: & 1.10100 & +13: & 0.01101 & \\ \hline & 0.00001 & & 0.00001 & \\ & \hookrightarrow\ 1 & & \hookrightarrow\ 1 & \text{Add in EAC} \\ \hline +2: & 0.00010 & +2: & 0.00010 & \end{array}$$

Figure 9.16

Addition

$$\begin{array}{rl} -13: & 1.10011 \\ -11: & 1.10101 \\ \hline -24: & 1.01000 \end{array}$$

$$\begin{array}{rlrl} +13: & 0.01101 & -13: & 1.10011 \\ -11: & 1.10101 & +11: & 0.01011 \\ \hline +2: & 0.00010 & -2: & 1.11110 \end{array}$$

$$\begin{array}{rlrl} -11: & 1.10101 & +11: & 0.01011 \\ +13: & 0.01101 & -13: & 1.10011 \\ \hline +2: & 0.00010 & -2: & 1.11110 \end{array}$$

Subtraction

$$\begin{array}{rlrl} +13: & 0.01101 & +13: & 0.01101 \\ (-11): & 1.10101 & -(+11): & 0.01011 \\ & & & \\ & 0.01101 & & 0.01101 \\ & 0.01010 & & 1.10100 \\ & 1 & & 1 \\ \hline +24: & 0.11000 & +2: & 0.00010 \end{array}$$

Fig. 9.17. 2's complement addition and subtraction.

the same as the statement "change the sign of the subtrahend and add." The conversion to the subtrahend 2's complement is initiated by taking the 1's complement, and completed by the addition of 1 at the LSB.

Comparison of Notations. For addition and subtraction the 2's complement is considered to be the most straightforward in implementation, with the 1's complement next best, and sign and magnitude most complex.

Exactly one cycle is required for addition or subtraction of any 2's complement numbers. This seems to give a definite advantage to 2's complement notation, but it creates problems in multiplication and division. Sign and magnitude representation, however, is very straightforward in performing multiplication and division.

A disadvantage of 1's complement notation is that an EAC (end-around-carry) may require additional time for carry propagation.

To illustrate the difference resulting from the use of a particular notation, Figs. 9.18, 9.19, and 9.20 show simple adders/subtracters using each of the representations.

9.6 ADDITION AND SUBTRACTION OF DECIMAL NUMBERS WITH BINARY REPRESENTATIONS

In some instances it is desirable to perform arithmetic operations directly, using decimal numbers. This need often occurs in systems where the result of the arithmetic operation is to be displayed directly and it is desired to avoid conversions. Most numerical display systems interface directly with decimal code representations. Two codes will be considered here: the BCD (binary-coded decimal) and the excess-3 code. For this discussion, BCD will mean the 8-4-2-1 coding scheme. For BCD the binary numbers correspond exactly to the value of the decimal number which

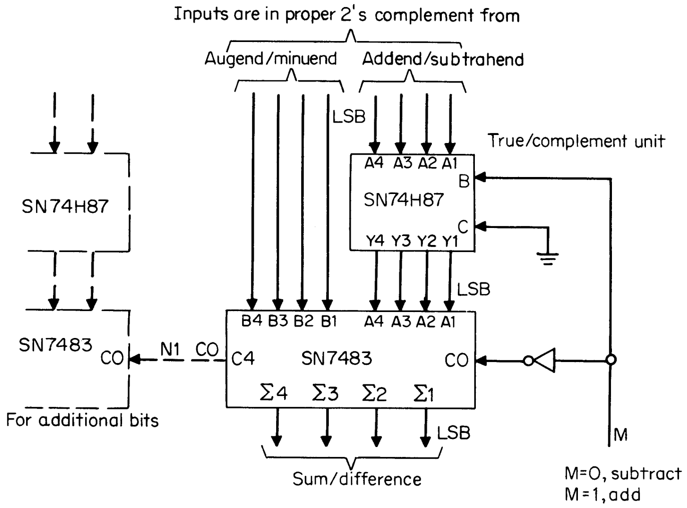

Inputs are in proper 2's complement from
Augend/minuend
Addend/subtrahend
LSB
True/complement unit
A4 A3 A2 A1
B
SN74H87
C
Y4 Y3 Y2 Y1
LSB
SN74H87
SN7483
CO
N1
CO
B4 B3 B2 B1 A4 A3 A2 A1
C4
SN7483
CO
Σ4 Σ3 Σ2 Σ1
For additional bits
LSB
M
Sum/difference
M=0, subtract
M=1, add

Fig. 9.18. Adder/subtracter using 2's complement notation; 4 bits (including sign-bit).

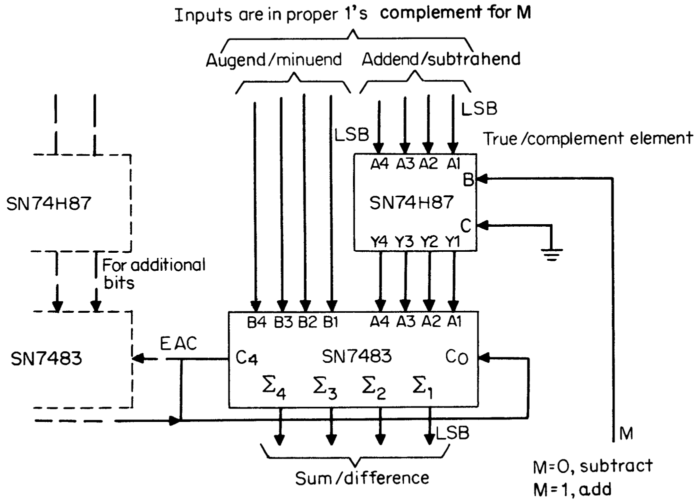
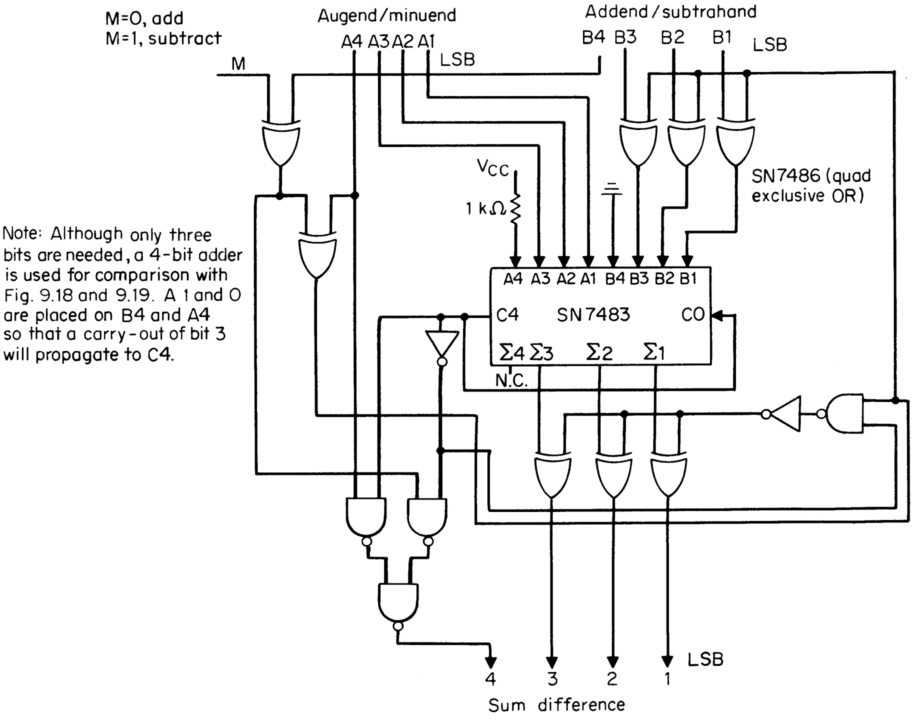

Inputs are in proper 1's complement for M
Augend/minuend
Addend/subtrahend
LSB
LSB
True/complement element
A4 A3 A2 A1
B
SN74H87
C
Y4 Y3 Y2 Y1
SN74H87
For additional bits
B4 B3 B2 B1
A4 A3 A2 A1
SN7483
EAC
C4
SN7483
Co
Σ4 Σ3 Σ2 Σ1
LSB
Sum/difference
M
M=0, subtract
M=1, add

Fig. 9.19. Adder/subtracter using 1's complement notation; 4 bits (including sign-bit).


M=0, add
M=1, subtract
M
Augend/minuend
A4 A3 A2 A1
LSB
Addend/subtrahand
B4 B3 B2 B1
LSB
SN7486 (quad exclusive OR)
Vcc
1 kΩ
A4 A3 A2 A1 B4 B3 B2 B1
C4
SN7483
CO
Σ4 Σ3
Σ2
Σ1
N.C.
4
3
2
1
LSB
Sum difference

Note: Although only three bits are needed, a 4-bit adder is used for comparison with Fig. 9.18 and 9.19. A 1 and 0 are placed on B4 and A4 so that a carry-out of bit 3 will propagate to C4.

Fig. 9.20. Adder/subtracter using sign and magnitude notation; 4 bits (including sign-bit).

Decimal	Binary coded decimal
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

10^3 10^2 10^1 10^0
Example: $(9835)_{10}$ = 1001/1000/0011/0101

Fig. 9.21. BCD representation.

Decimal	Uncorrected BCD sum $C_4\Sigma_4\Sigma_3\Sigma_2\Sigma_1$	Corrected BCD sum $C_4\Sigma_4\Sigma_3\Sigma_2\Sigma_1$	
0	0000	0000	No correction required
1	0001	0001	
2	0010	0010	
3	0011	0011	
4	0100	0100	
5	0101	0101	
6	0110	0110	
7	0111	0111	
8	1000	1000	
9	1001	1001	
10	1010	10000	Correction required
11	1011	10001	
12	1100	10010	
13	1101	10011	
14	1110	10100	
15	1111	10101	
16	10000	10110	
17	10001	10111	
18	10010	11000	
19	10011	11001	

Fig. 9.22. Results of BCD addition with corrections indicated.

is represented (Fig. 9.21). The advantage of this code is that addition is identical with ordinary binary addition for each character. The sum for each decimal digit must, however, be corrected to take care of carries to the next digit and results which have the binary values 10 through 15.

One of the disadvantages of the BCD system is that it is not easy to form its 9's complement. If the 1's complement of a 4-bit binary number is taken, the result in BCD is the 15's complement.

BCD Addition. When BCD addition is performed with a possible carry-in, 20 different sums can be produced. Of these, only 10 will be correct; the remainder will require correction.

For a sum equal to or greater than 10_{10} (subscript = base), a subtraction of 10_{10} will give the correct result for the digit in question, and a carry to the next decade will also be required, as can be seen in Fig. 9.22. The required subtraction of 10_{10} can be achieved by adding the 2's complement of the BCD representation of 10_{10}

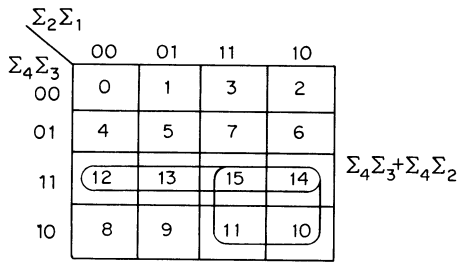


Figure 9.23

(1010_2), which is 0110_2 in BCD or decimal 6. A second 4-bit adder stage is used for the addition. A decoding scheme is required to generate a carry for the following digit, and to control when 6 is added to the sum for correction. The Karnaugh map in Fig. 9.23 represents this decoding of the uncorrected sums. The result of Fig. 9.23 is combined with the carry-out signal to detect a possible sum of 10_{10} through 19_{10}, thus giving

$$C_n = C_4 + \Sigma_4\Sigma_3 + \Sigma_4\Sigma_2 \tag{16}$$

which is implemented in Fig. 9.24.

BCD Subtraction. Figure 9.25 shows an algorithm for a BCD subtracter. The 1's complement of the BCD representation of the subtrahend is entered into adder 1, and the true or complement of the result is transferred to adder 2, where either a 1010 or 0000 is added, depending on the sign of the decade in question and the sign of the total result. Examples of a positive and a negative total result are shown

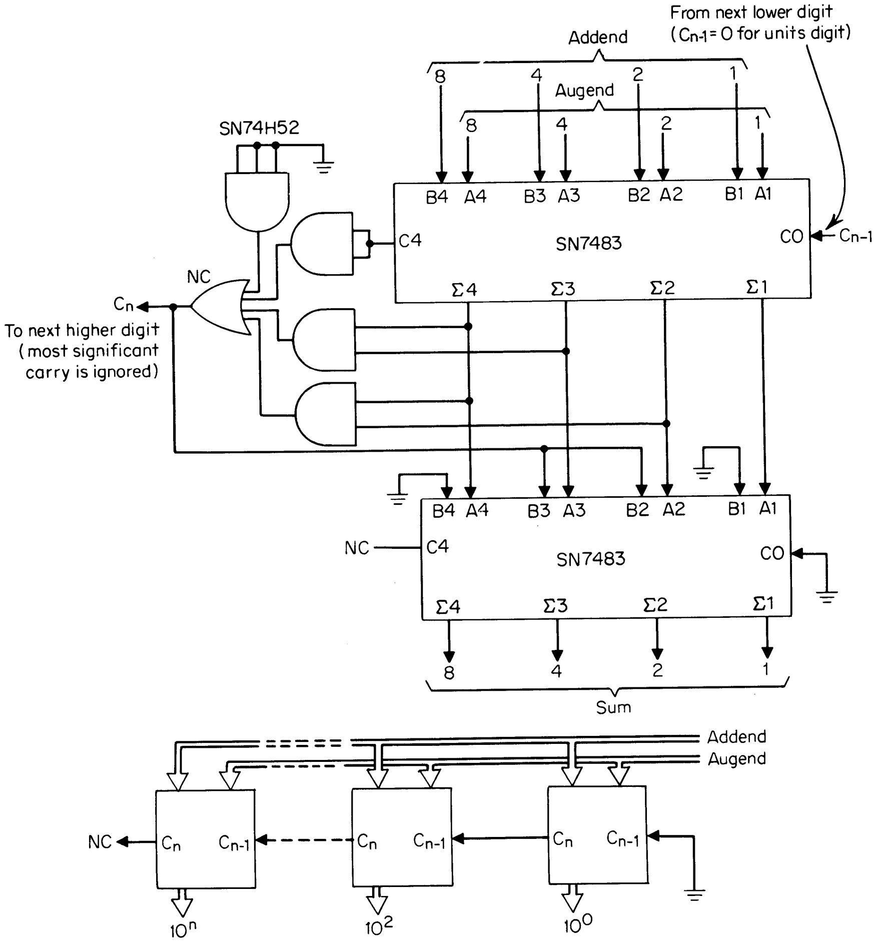
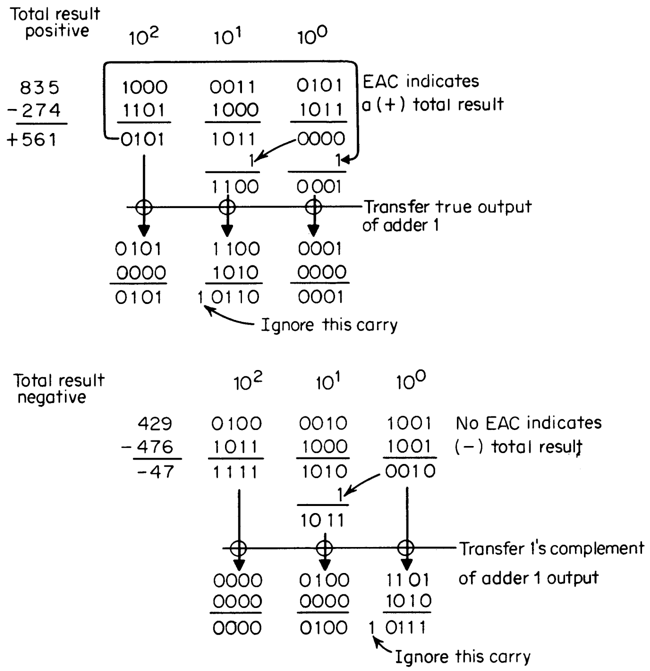

From next lower digit
(Cn-1 = 0 for units digit)
Addend
8
4
2
1
Augend
8
4
2
1
SN74H52
B4 A4 B3 A3 B2 A2 B1 A1
C4
SN7483
CO
Cn-1
Σ4 Σ3 Σ2 Σ1
NC
Cn
To next higher digit
(most significant
carry is ignored)
B4 A4 B3 A3 B2 A2 B1 A1
NC
C4
SN7483
CO
Σ4 Σ3 Σ2 Σ1
8
4
2
1
Sum
Addend
Augend
NC
Cn
Cn-1
Cn
Cn-1
Cn
Cn-1
10^n
10^2
10^0

Fig. 9.24. BCD adder.


Total result positive
10^2
10^1
10^0
835
-274
+561
1000
1101
0101
0011
1000
1011
1
1100
0101
1011
0000
1
0001
EAC indicates a (+) total result
Transfer true output of adder 1
0101
0000
0101
1100
1010
1 0110
0001
0000
0001
Ignore this carry
Total result negative
10^2
10^1
10^0
429
-476
-47
0100
1011
1111
0010
1000
1010
1
1011
1001
1001
0010
No EAC indicates (-) total result
Transfer 1's complement of adder 1 output
0000
0000
0000
0100
0000
0100
1101
1010
1 0111
Ignore this carry

Decade result	Sign of total result	
	(+) EAC = 1	(−) EAC = 0
(+) $C_n = 1$	Transfer true results of adder 1 0000 added in adder 2	Transfer 1's complement of result of adder 1 1010 added in adder 2
(−) $C_n = 0$	1010 added in adder 2	0000 added in adder 2

Algorithm for BCD subtracter

Fig. 9.25. BCD subtracter: examples and algorithm.

in Fig. 9.25. Arrows indicate EAC (end-around-carry) or carry to the next decade.

Figure 9.26 shows a subtracter using the 1's complement of the subtrahend method. To handle more than seven decades, additional inverters must be used for the E_i signal. When the minuend and subtrahend are equal, a minus zero is the indicated result. This may or may not be a disadvantage, depending on the overall system. It may be useful in some applications to use a conditional true/complement (SN7486 quad exclusive-OR package) or both A and B inputs to adder 1; these units can then be controlled so that the subtrahend and minuend functions can be interchanged, thus avoiding a data transfer to the fixed function inputs shown in Fig. 9.26. If negative subtraction results are not required for a particular system, the five exclusive-OR gates can be deleted, and an inverter connected from the $C4$

terminal of adder 1 with its output to terminals *B*4 and *B*2 of adder 2. The *E* signals are also deleted, but the EAC (end-around-carry) connection is maintained. With this system, the absence of an EAC would mean either a zero result (which is in 15's complement form, all 1s) or a negative result; for this system, that would be an error.

Another method for performing BCD subtraction is the addition of the 9's complement of the subtrahend to the minuend.

Figure 9.27 indicates the decoding required for conversion to BCD 9's complement representation.

Two possibilities are shown for implementation of the logic of Fig. 9.27. The first (Fig. 9.28) uses a single 4-bit adder and four inverters; the second method (Fig. 9.29) may be more economical for a large number of decades where the packages required can be more fully utilized.

A BCD 9's complement subtracter stage is made up of a BCD adder stage (Fig. 9.24) and a 9's complement stage (Fig. 9.28 or 9.29). Figure 9.30 shows a multibit

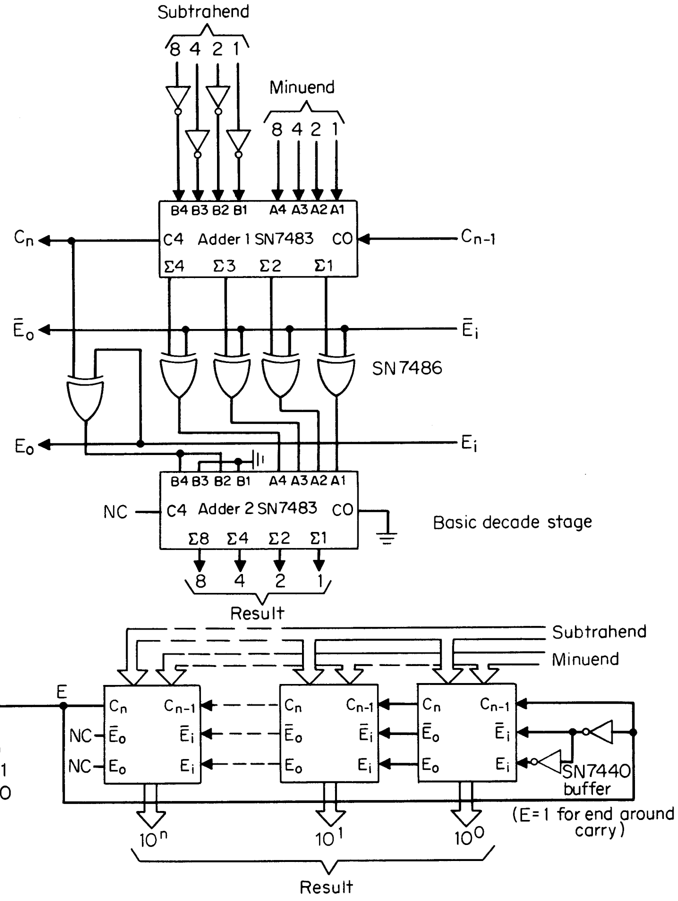

Subtrahend
8 4 2 1
Minuend
8 4 2 1
B4 B3 B2 B1 A4 A3 A2 A1
C4 Adder 1 SN7483 CO
Σ4 Σ3 Σ2 Σ1
Cn
Cn-1
Ē0
Ēi
SN7486
E0
Ei
B4 B3 B2 B1 A4 A3 A2 A1
NC
C4 Adder 2 SN7483 CO
Σ8 Σ4 Σ2 Σ1
Basic decade stage
8 4 2 1
Result
Subtrahend
Minuend
E
Cn Cn-1
NC Ē0 Ēi
NC E0 Ei
Sign
(+) if E=1
(−) if E=0
SN7440 buffer
(E=1 for end around carry)
10^n
10^1
10^0
Result

Fig. 9.26. BCD subtracter: 1's complement of subtrahend.

True		9's Complement	
Dec.	BCD 8 4 2 1 A B C D	Dec.	BCD 8 4 2 1 W X Y Z
0	0 0 0 0	9	1 0 0 1
1	0 0 0 1	8	1 0 0 0
2	0 0 1 0	7	0 1 1 1
3	0 0 1 1	6	0 1 1 0
4	0 1 0 0	5	0 1 0 1
5	0 1 0 1	4	0 1 0 0
6	0 1 1 0	3	0 0 1 1
7	0 1 1 1	2	0 0 1 0
8	1 0 0 0	1	0 0 0 1
9	1 0 0 1	0	0 0 0 0

$W = \bar{A}\,\bar{B}\,\bar{C}$
$X = B \oplus C$
$Y = C$
$Z = \bar{D}$

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

Z

$Z = \bar{D}$

AB \ CD	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

W

$W = \bar{A}\bar{B}\bar{C}$

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

X

$X = B\bar{C} + \bar{B}C = B \oplus C$

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	X	X	X	X
10	0	0	X	X

Y

$Y = C$

Fig. 9.27. BCD 9's complement decoding.

BCD 9's complement subtracter. The 9's complement subtraction for a positive result produces a carry-out of the MSD (most significant digit), and the digit results are represented as their true values.

A negative result will not produce a carry from the MSD, and the digits will be represented in 9's complement form. If necessary, 9's complement generator stages can be used to convert a negative result to the true form. Where the following operation is an addition of this negative result to another number (an actual subtraction if the other number is positive), then this result can go directly into a BCD adder, bypassing the 9's complement generator. For display of a negative result, conversion to the true value is required if standard BCD decoding is to be used.

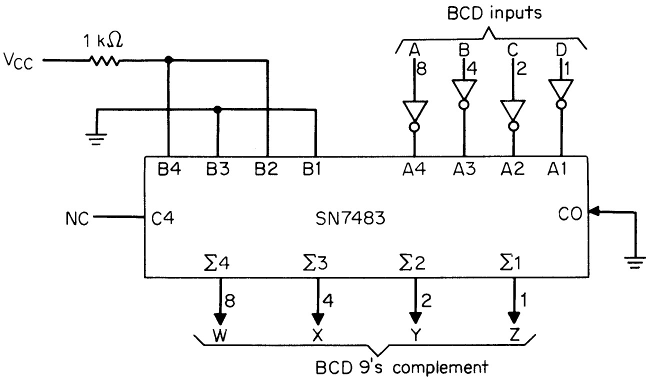


Fig. 9.28. BCD 9's complement generator (method 1).

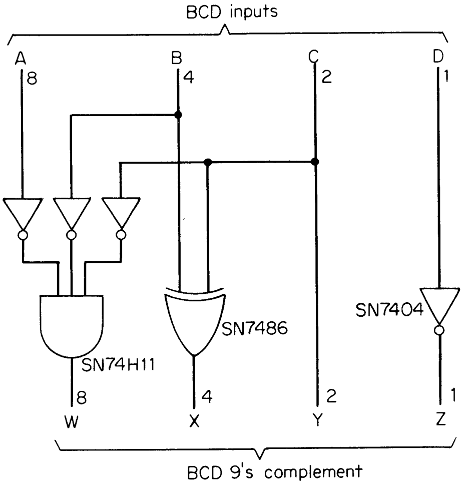

BCD inputs
A
8
B
4
C
2
D
1
SN74H11
SN7486
SN7404
8
4
2
1
W
X
Y
Z
BCD 9's complement

Fig. 9.29. BCD 9's complement generator (method 2).

The BCD adder/subtractor shown in Fig. 9.31 is a combination of the adder of Fig. 9.24 and the subtracter of Fig. 9.26. The additional logic indicated responds to the *M* (mode control) signal to set up the circuit as either an adder or a subtracter. The same comments previously made on BCD addition and subtraction apply to the combined adder/subtracter. If more than four decades are to be handled, the inverter driving the $\bar{E}_i$ input must be changed. Another SN7404 may be paralleled, or an SN7440 buffer gate can be used, depending on the number of decades.

Excess-3 Decimal Code. To avoid the complexity of obtaining the 9's complement with the BCD (8-4-2-1) representation, the excess-3 code can be used. This code gives the 9's complement when the individual bits are inverted—equivalent to taking the 1's complement from a binary point of view. There are other codes with this property, but the excess-3 code is the most generally accepted. Figure 9.32 demonstrates that the excess-3 code is formed by adding 3 to each BCD decimal digit representation.

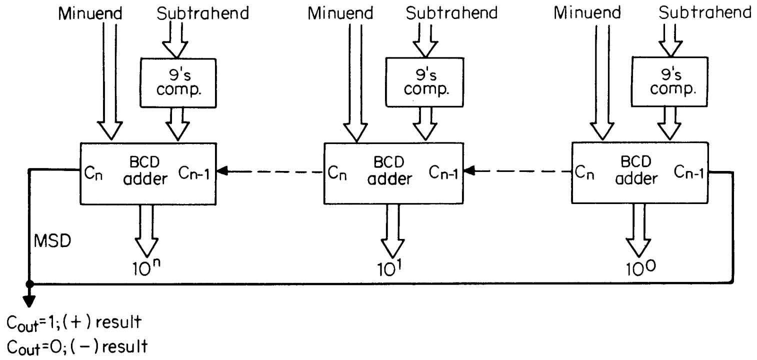

Minuend
Subtrahend
9's comp.
BCD adder
Cn
Cn-1
MSD
10^n
10^1
10^0
Cout=1; (+) result
Cout=0; (−) result

Fig. 9.30. BCD subtracter: subtrahend 9's complement.

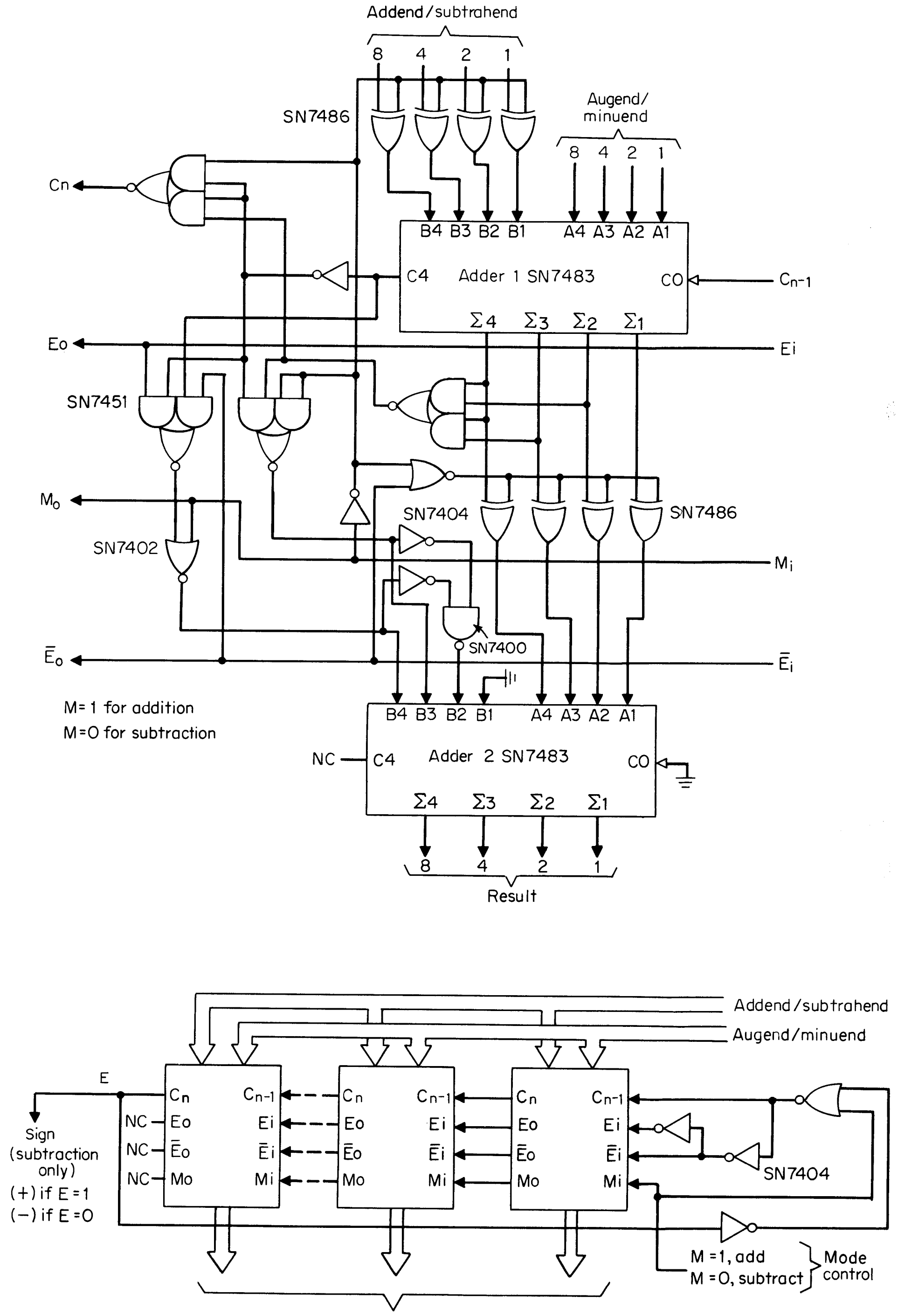

Addend/subtrahend
8
4
2
1
SN7486
Augend/
minuend
8
4
2
1
Cn
B4 B3 B2 B1
A4 A3 A2 A1
C4
Adder 1 SN7483
CO
Cn-1
Σ4
Σ3
Σ2
Σ1
Eo
Ei
SN7451
Mo
SN7404
SN7486
SN7402
Mi
SN7400
Ēo
Ēi
M=1 for addition
M=0 for subtraction
B4 B3 B2 B1
A4 A3 A2 A1
NC
C4
Adder 2 SN7483
CO
Σ4
Σ3
Σ2
Σ1
8
4
2
1
Result
Addend/subtrahend
Augend/minuend
E
Sign
(subtraction
only)
(+) if E=1
(−) if E=0
NC
NC
NC
Cn
Eo
Ēo
Mo
Cn-1
Ei
Ēi
Mi
SN7404
M=1, add
M=0, subtract
Mode
control
Result

Fig. 9.31. BCD adder/subtracter: 1's complement of subtrahend.

Decimal	Excess-3
0	0011
1	0100
2	0101
3	0110
4	0111
5	1000
6	1001
7	1010
8	1011
9	1100

Figure 9.32

Addition is performed as follows: For a no-carry, a 3 must be subtracted from the sum. This will be accomplished by adding 13, which is the 2's complement of the binary representation of 3; the carry-out for this addition is disregarded. For a carry-out, the effect is a carry-out of magnitude 16, which is 6 too much. A magnitude of 6 must be added and 3 must be subtracted to obtain an excess-3 sum for this decade; the net effect is an addition of 3.

Refer to Fig. 9.33 for examples of excess-3 addition and subtraction. Notice that sub-

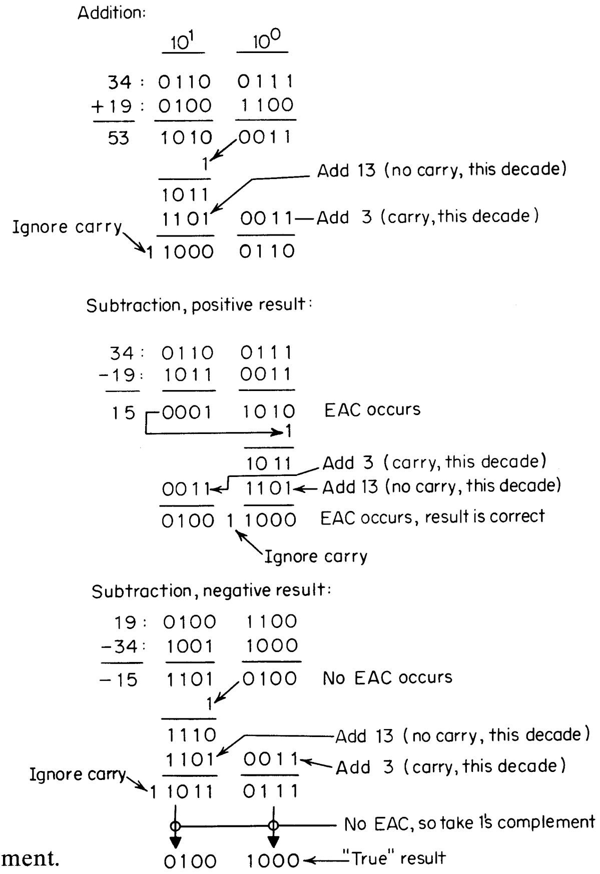

Addition:
10^1
10^0
34 : 0110 0111
+19 : 0100 1100
53 1010 0011
1
Add 13 (no carry, this decade)
1011
Ignore carry
1101
0011—Add 3 (carry, this decade)
1 1000 0110
Subtraction, positive result:
34: 0110 0111
-19: 1011 0011
15 0001 1010
EAC occurs
1
1011
Add 3 (carry, this decade)
0011
1101
Add 13 (no carry, this decade)
0100 1 1000
EAC occurs, result is correct
Ignore carry
Subtraction, negative result:
19: 0100 1100
-34: 1001 1000
-15 1101 0100
No EAC occurs
1
1110
Add 13 (no carry, this decade)
Ignore carry
1101
0011
Add 3 (carry, this decade)
1 1011 0111
No EAC, so take 1's complement
0100 1000
"True" result

Fig. 9.33. Excess-3 addition and subtraction: subtrahend 9's complement.

traction is performed by taking the decimal 9's complement (binary 1's complement) of the subtrahend and adding it to the minuend. A positive result from subtraction gives the true value while a negative result is in 9's complement form. For negative results, the 9's complement of the output is taken to obtain the true value.

Figure 9.34 shows an excess-3 adder/subtracter with an additional true/complement element so that the true value is always obtained. For subtraction with equal operands, the result is a negative zero, which is the same as for the BCD subtracter. If only an excess-3 adder is required, the two true/complement units and the gating at the LSD (least significant digit) can be deleted, and the C_n of the LSD can be tied to logical 0. If only a subtraction operation is required, the true/complement unit for the subtrahend input to adder 1 can be replaced with four inverters; the E output of the MSD is tied directly to the C_{n-1} and E_i of the LSD stage. For this circuit arrangement, 10 decades are maximum; beyond 10, the external gating at the LSD will have to be changed to allow for a larger fan-out.

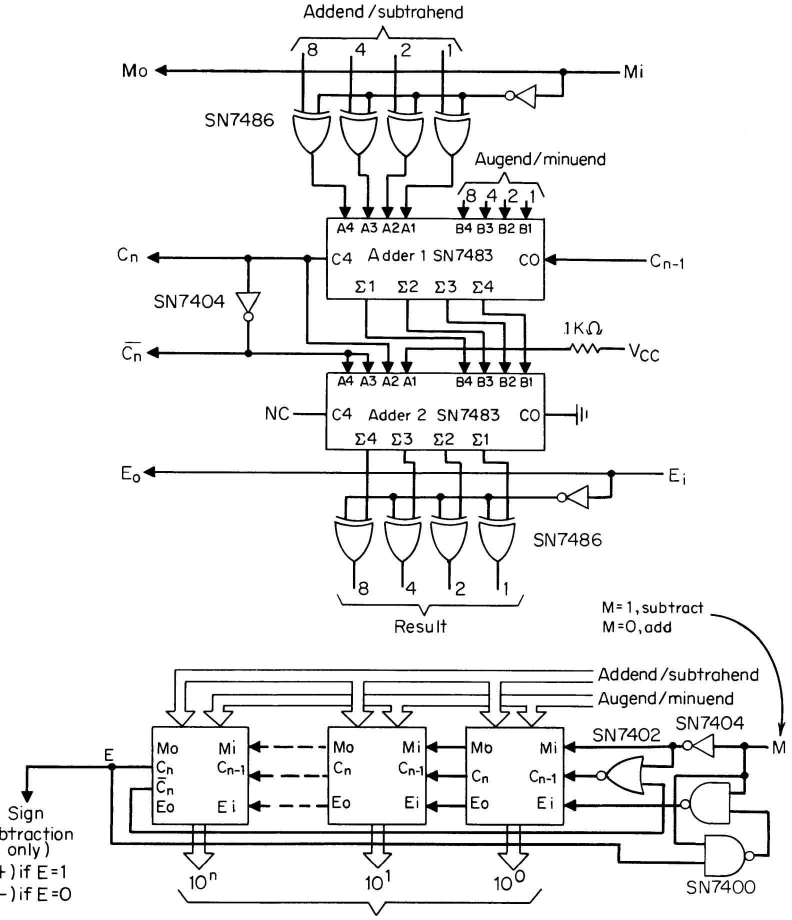

Addend / subtrahend
8 4 2 1
Mo
Mi
SN7486
Augend/minuend
8 4 2 1
A4 A3 A2 A1
B4 B3 B2 B1
C4 Adder 1 SN7483 CO
Σ1 Σ2 Σ3 Σ4
Cn
Cn-1
SN7404
C̄n
1KΩ
Vcc
A4 A3 A2 A1
B4 B3 B2 B1
NC
C4 Adder 2 SN7483 CO
Σ4 Σ3 Σ2 Σ1
Eo
Ei
SN7486
8 4 2 1
Result
M=1, subtract
M=0, add
Addend/subtrahend
Augend/minuend
SN7404
SN7402
M
E
Mo Mi
Cn Cn-1
C̄n
Eo Ei
Mo Mi
Cn Cn-1
Eo Ei
Mo Mi
Cn Cn-1
Eo Ei
Sign
(subtraction only)
(+) if E=1
(−) if E=0
10ⁿ
10¹
10⁰
SN7400

Figure 9.34

For the decimal adders and subtracters illustrated, a sign and magnitude notation is implied. Actually, it is even more restrictive in that the sign of the operands is assumed to be positive. A system of using sign and magnitude, 9's complement, or 10's complement notation could be used. It would follow in a general way the sign and magnitude, 1's complement, or 2's complement notation for straight binary systems previously discussed. The rules for signed decimal arithmetic systems would be more complex than for straight binary. In some cases where a general signed decimal arithmetic system is being considered, it would be better to use a signed binary system and to convert between the decimal and binary representations.

9.7 FAST BINARY ADDITION

As the word length of a typical carry-propagating parallel adder increases, the time required to complete an addition increases by the delay time per stage for each bit added. There are several techniques for speeding up the process, such as carry look-ahead, asynchronous (carry completion) addition, and conditional sum addition.† Among these, only carry look-ahead techniques will be considered here.

Carry Look-ahead Adders. To facilitate the discussion to follow, three new symbols are defined:

$$^{1}G = AB \tag{17}$$

$$^{0}G = \overline{AB} \tag{18}$$

$$P = A\bar{B} + \bar{A}B = A \oplus B \tag{19}$$

The A and B symbols represent a pair of operand bits to any stage within an adder. The ^{1}G is the symbol for a 1-carry generation; it is produced only when both A and B are 1. For this case, a carry-out is always generated regardless of the carry input to the stage. The ^{0}G represents the case of a 0-carry-out of the stage in question. For both A and B equal to 0, the carry-out is 0 regardless of the input carry. P is the symbol for propagate, which means that if this condition exists ($A = 0$, $B = 1$ or $A = 1$, $B = 0$), then whatever state the carry input has will be propagated through as the carry-out of this stage. For a 1-carry, it can be stated as

$$^{1}C_n = {^{1}G_n} + P_n\,{^{1}C_{n-1}} \tag{20}$$

where the $^{1}C_n$ stands for a 1-carry output, and the $^{1}C_{n-1}$ stands for a 1-carry input from the previous stage. A no-carry (or 0-carry) condition can be stated as

$$^{0}C_n = {^{0}G_n} + P_n\,{^{0}C_{n-1}} \tag{21}$$

where the $^{0}C_n$ represents a 0-carry output, and $^{0}C_{n-1}$ represents a 0-carry input from the previous stage. For a 1-carry out of the kth stage of an adder, the following can be stated:

$$C_k = G_k + P_k C_{k-1} \tag{22}$$

---

†For full discussions of these techniques, see Ivan Flores, "The Logic of Computer Arithmetic," Prentice-Hall, Inc., Englewood Cliffs, N.J., 1963.

The pre-superscripts are dropped from here on and will be considered equal to 1. Substituting in Eq. (22) for G_k and P_k gives

$$C_k = A_kB_k + (\bar{A}_kB_k + A_k\bar{B}_K)C_{k-1} \tag{23}$$

This reduces to

$$C_k = A_kB_k + B_kC_{k-1} + A_kC_{k-1} \tag{24}$$

which is the same as Eq. (5). Refer to the Karnaugh map of Fig. 9.4 for insight into the reduction. Eq. (24) can be written as

$$C_k = A_kB_k + C_{k-1}(A_k + B_k) \tag{25}$$

This indicates that the P_k term can be the inclusive-OR function of A_k and B_k [Eq. (25)] as well as the previously defined exclusive-OR of Eq. (19). This property can be used to reduce the number of components required in the hardware implementation of this function. The carry-out of stage $k - 1$ is

$$C_{k-1} = G_{k-1} + P_{k-1}C_{k-2} \tag{26}$$

Now if Eq. (26) is substituted for C_{k-1} in Eq. (22), the following will be obtained:

$$\begin{aligned} C_k &= G_k + P_k(G_{k-1} + P_{k-1}C_{k-2}) \qquad (27)\\ &= G_k + P_kG_{k-1} + P_kP_{k-1}C_{k-2} \end{aligned}$$

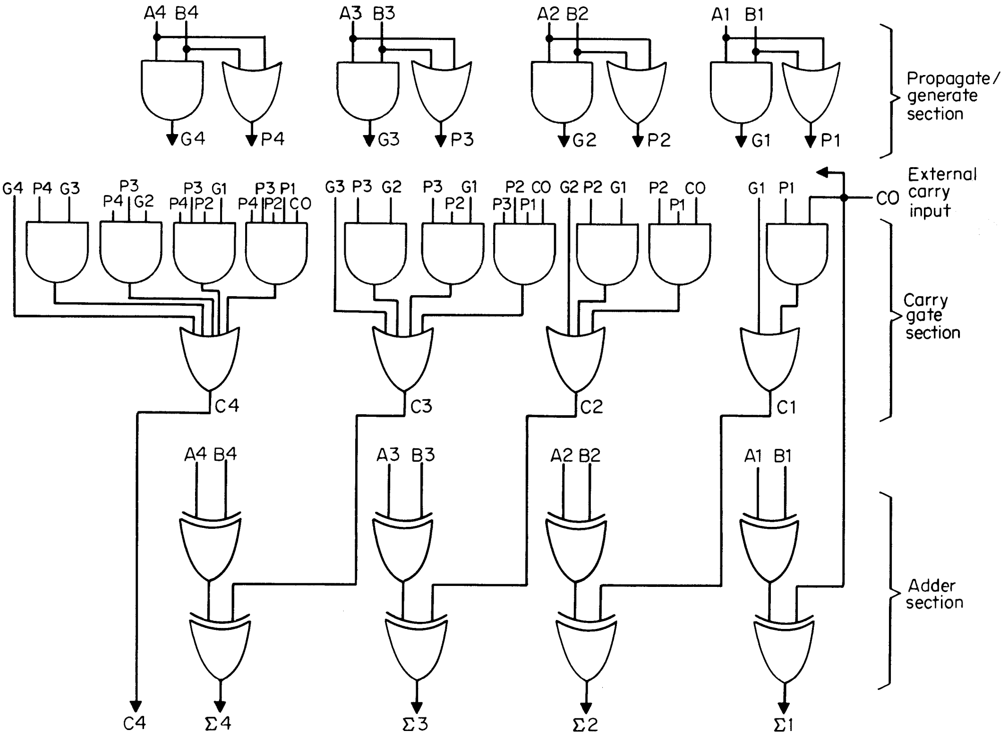
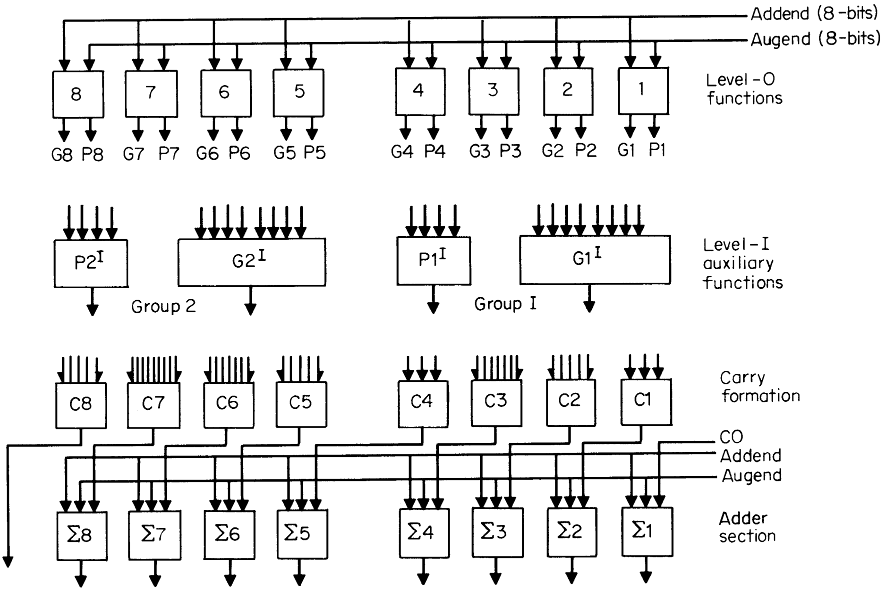

A4 B4
A3 B3
A2 B2
A1 B1
G4
P4
G3
P3
G2
P2
G1
P1
Propagate/ generate section
G4 P4 G3 P3 P4 G2 P3 G1 P4 P2 P3 P1 P4 P2 CO
G3 P3 G2 P3 G1 P2 P2 CO P3 P1
G2 P2 G1 P2 CO P1
G1 P1
CO
External carry input
Carry gate section
C4
C3
C2
C1
A4 B4
A3 B3
A2 B2
A1 B1
Adder section
C4
Σ4
Σ3
Σ2
Σ1

Fig. 9.35. 4-bit 0-level carry look-ahead adder.


Addend (8-bits)
Augend (8-bits)
8
7
6
5
4
3
2
1
Level-0 functions
G8 P8 G7 P7 G6 P6 G5 P5 G4 P4 G3 P3 G2 P2 G1 P1
P2^I
G2^I
P1^I
G1^I
Group 2
Group I
Level-I auxiliary functions
C8
C7
C6
C5
C4
C3
C2
C1
Carry formation
CO
Addend
Augend
Σ8
Σ7
Σ6
Σ5
Σ4
Σ3
Σ2
Σ1
Adder section

$C1 = G1 + P1\,CO$

$C2 = G2 + P2\,G1 + P2\,P1\,CO$

$C3 = G3 + P3\,G2 + P3\,P2\,G1 + P3\,P2\,P2\,CO$

$C4 - G1^{I} + P1^{I}\,CO$

$C5 = G5 + P5\,G1^{I} + P5\,P1^{I}\,CO$

$C6 = G6 + P6\,G5 + P6\,P5\,G1^{I} + P6\,P5\,G1^{I} + P6\,P5\,P1^{I}\,CO$

$C7 = G7 + P7\,G6 + P7\,P6\,G5 + P7\,P6\,P5\,G1^{I} + P7\,P6\,P5\,P1^{I}\,CO$

$C8 = G2^{I} + P2^{I}\,G1^{I} + P2^{I}\,P1^{I}\,CO$

$G1^{I} = G4 + P4\,G3 + P4\,P3\,G2 + P4\,P3\,P2\,G1$

$P1^{I} = P4\,P3\,P2\,P1$

$G2^{I} = G8 + P8\,G7 + P8\,P7\,G6 + P8\,P7\,P6\,G5$

$P2^{I} = P8\,P7\,P6\,P5$

Fig. 9.36. 8-bit carry look-ahead adder with first-level auxiliary functions.

so that in general it can be seen that

$$C_k = G_k + P_kG_{k-1} + P_kP_{k-1}G_{k-2} + \cdots + P_kP_{k-1}P_{k-2}\cdots P_1C_0 \qquad (28)$$

where C_0 is an external carry input.

An illustrative example of a 0-level carry look-ahead system is shown in Fig. 9.35. For comparison, a 4-bit ripple-carry parallel adder making use of the logic of Fig. 9.5 would have a path length of eight levels from the external carry input C_0 to the C_4 output. The carry look-ahead adder of Fig. 9.35 has a path length of three levels from C_0 to C_4. This difference becomes much more significant as the word length becomes longer. The ripple-carry adder gains an additional two levels for each bit, whereas the carry look-ahead maintains the three-level path lengths. As the word length of the carry look-ahead adder is increased, it very quickly requires a massive number of gates and creates fan-in and fan-out problems.

A full 0-level carry look-ahead adder for long words is in most cases impractical. The use of FLA (first-level auxiliary) functions (also known as two-level look-ahead) is a method for trading off some of the high-speed capability of the full 0-level look-ahead for a reduction in hardware. Figure 9.36 is an illustrative example of an adder with FLA. For this example, groups of 4 bits make up the FLA functions; the 0-level functions are generated as in Fig. 9.35.

The effect of using FLA functions is shown in the reduction in complexity of the C_4 and C_8 terms. It is instructive to write out the C_8 term using only 0-level functions:

$$C_8 = G_8 + P_8G_7 + P_8P_7G_6 + P_8P_7P_6G_5 + P_8P_7P_6P_5G_4 + P_8P_7P_6P_5P_4G_3 + P_8P_7P_6P_5P_4P_3G_2 + P_8P_7P_6P_5P_4P_3P_2G_1 + P_8P_7P_6P_5P_4P_3P_2P_1C_0 \quad (29)$$

Now the C_8 term from Fig. 9.36 is written with FLA functions substituted:

$$C_8 = (G_8 + P_8G_7 + P_8P_7G_6 + P_8P_7P_6G_5) + (P_8P_7P_6P_5) \cdot (G_4 + P_4G_3 + P_4P_3G_2 + P_4P_3P_2G_1) + (P_8P_7P_6P_5)(P_4P_3P_2P_1)(C_0) \quad (30)$$

The equivalence with Eq. (29) is now easily recognized. Generalized equations for the FLA functions are

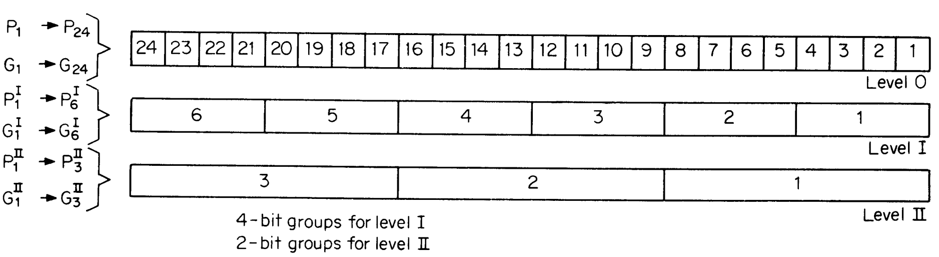
$$G^{I}_{k/g} = G_k + P_kG_{k-1} + \cdots + P_kP_{k-1} \cdots P_{k-g+2}G_{k-g+1} \quad (31)$$

$$P^{I}_{k/g} = P_kP_{k-1} \cdots P_{k-g} + 2P_{k-g+1} \quad (32)$$

where K = group number
g = number of bits within a group
$k/g = k \div g$

The next step in the same direction as FLA functions is the use of SLA (second-level auxiliary) functions. An example is shown in Fig. 9.37 with a block representation to indicate the group divisions.

SLA functions are generated from the FLA functions. The same form of equation as Eqs. (31) and (32) applies, but with the pre-superscript incremented by 1 on both sides of the equation. FLA functions provide a speed advantage approximately proportional to the group size, and the use of SLA functions provides an advantage which is approximately proportional to the square of the group size.



$$C_{14} = G_{14} + P_{14}G_{13} + P_{14}P_{13}G_3 + P_{14}P_{13}P^{I}_3G^{II}_1 + P_{14}P_{13}P_3P_1C_0$$

$$C_{22} = G_{22} + P_{22}G_{21} + P_{22}P_{21}G^{I}_5 + P_{22}P_{21}P^{I}_5G^{II}_2 + P_{22}P_{21}P^{I}_5P^{II}_2G^{II}_1 + P_{22}P_{21}P^{I}_5P^{II}_2P^{II}_1C_0$$

$$G^{II}_2 = G^{I}_4 + P^{I}_4G^{I}_3$$

Fig. 9.37. 24-bit carry look-ahead adder with second-level auxiliary functions.

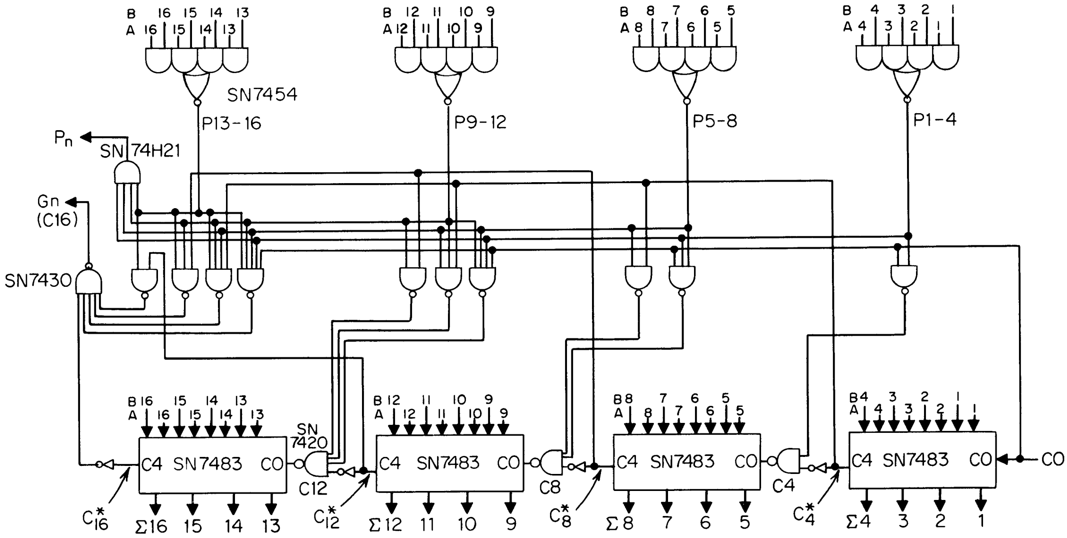

B 16 15 14 13
A 16 15 14 13
SN7454
P13-16
B 12 11 10 9
A 12 11 10 9
P9-12
B 8 7 6 5
A 8 7 6 5
P5-8
B 4 3 2 1
A 4 3 2 1
P1-4
Pn
SN 74H21
Gn
(C16)
SN7430
SN
7420
C4 SN7483 CO
C12
C8
C4
CO
C*16
C*12
C*8
C*4
Σ16 15 14 13
Σ12 11 10 9
Σ8 7 6 5
Σ4 3 2 1

Fig. 9.38. 16-bit adder with combination carry look-ahead and ripple carry.

Further levels of auxiliary functions can be used. The same process of grouping the previous level, as shown for FLA and SLA functions, is followed. Another possibility is a combination of carry look-ahead for groups using ripple carry. Figure 9.38 shows such an adder using the SN7483 high-speed 4-bit ripple-carry adder element. This is an efficient arrangement considering that a carry will ripple only through the last three stages of one SN7483 in the worst case. Equations for each type of carry are listed here to clarify Fig. 9.38:

$$C_4 = C_4^* + P_1P_2P_3P_4C_0 \quad \text{or} \quad C_4^* + P_{1\text{-}4}C_0 \tag{33}$$

$$C_8 = C_8^* + P_{5\text{-}8}C_4^* + P_{1\text{-}8}C_0 \tag{34}$$

$$C_{12} = C_{12}^* + P_{9\text{-}12}C_8^* + P_{5\text{-}12}C_4^* + P_{1\text{-}12}C_0 \tag{35}$$

$$C_{16} = G_n = C_{16}^* + P_{12\text{-}16}C_{12}^* + P_{9\text{-}16}C_8^* + P_{5\text{-}16}C_4^* + P_{1\text{-}16}C_0 \tag{36}$$

$$P_n = P_{1\text{-}16} \tag{37}$$

The G_n and P_n terms are used when additional word length is needed. A 32-bit arrangement is shown in Fig. 9.39, using the basic circuit of Fig. 9.38. Additional word length could also be achieved by a continuation of the processes of Fig. 9.38. The addition of three more SN7483 elements (for a total of 28 bits) could be accommodated by the SN7430 8-input NAND gate. The P_n term (if needed) would now require a NAND gate and an inverter (two levels) because of the fan-in.

There are many useful combinations other than those shown here of multilevel carry look-ahead and look-ahead combined with ripple carry. Each design is a special case determined by the requirements of word length, speed, cost, logic product line, and many other possible requirements.

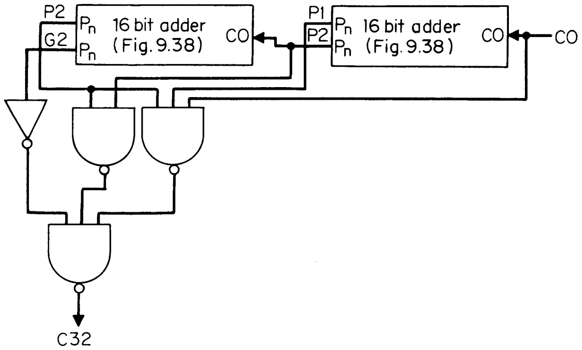

P2
G2
Pn
Pn
16 bit adder
(Fig. 9.38)
CO
P1
P2
Pn
Pn
16 bit adder
(Fig. 9.38)
CO
CO
C32

Fig. 9.39. 32-bit adder with carry look-ahead between 16-bit groups.

9.8 ADDER APPLICATIONS TO BINARY NUMBER REPRESENTATION CONVERSION

The 4-bit adder SN7483 can be applied to several of the commonly used binary code number conversion circuits.

Serial Binary to Parallel BCD Converter. The serial binary to parallel BCD converter of Fig. 9.40 operates by adding 0 for numbers 0 through 4 and adding 3 before shifting if it contains 5, 6, 7, 8, or 9. States 10 through 15 never occur if the above corrections are applied. Initially the registers must be cleared by taking the mode control *M* low for four clock pulses and then returning it to a high state.

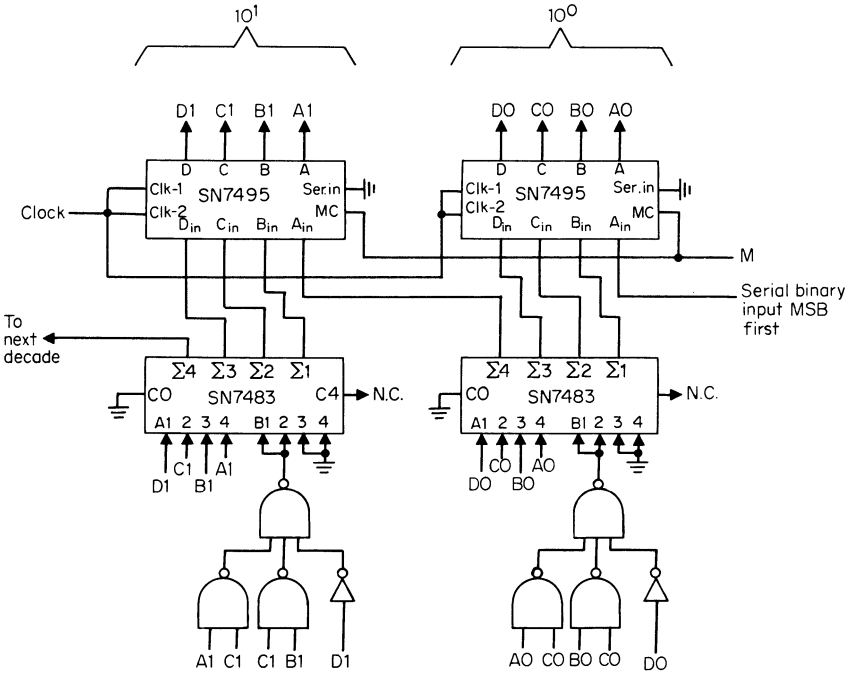

10^1
10^0
D1 C1 B1 A1
D0 C0 B0 A0
D C B A
Clk-1
Clk-2
SN7495
Ser.in
MC
D_{in} C_{in} B_{in} A_{in}
Clock
M
Serial binary input MSB first
To next decade
Σ4 Σ3 Σ2 Σ1
CO
SN7483
C4
N.C.
A1 2 3 4 B1 2 3 4
D1 C1 B1 A1
D0 C0 B0 A0
A1 C1 C1 B1 D1
A0 C0 B0 C0 D0

Fig. 9.40. Serial binary to parallel BCD converter (two decades shown).

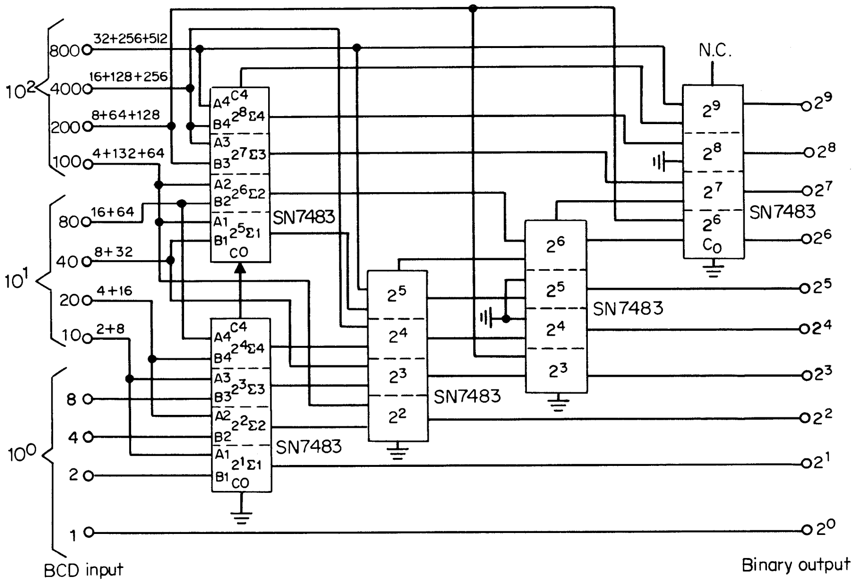

32+256+512
8000
16+128+256
10^2
4000
8+64+128
2000
4+132+64
1000
16+64
800
8+32
400
10^1
4+16
200
2+8
100
8
4
10^0
2
1
BCD input
SN7483
SN7483
SN7483
SN7483
N.C.
2^9
2^8
2^7
2^6
2^5
2^4
2^3
2^2
2^1
2^0
Binary output

Fig. 9.41. Parallel BCD to binary converter.

The actual conversion requires n clock pulses for an n-bit word, where n = number of bits.

Parallel BCD to Parallel Binary Converter. The parallel BCD to binary converter of Fig. 9.41 uses straight combinational logic to achieve a high conversion speed. The conversion time is limited only by the propagation delay of the adders and is approximately 250 ns. The basic circuit can be extended to any number of decades.

Serial BCD to Serial Binary Converter. Figure 9.42 shows another use for the 4-bit adder in a serial BCD to serial binary converter. The three BCD decades

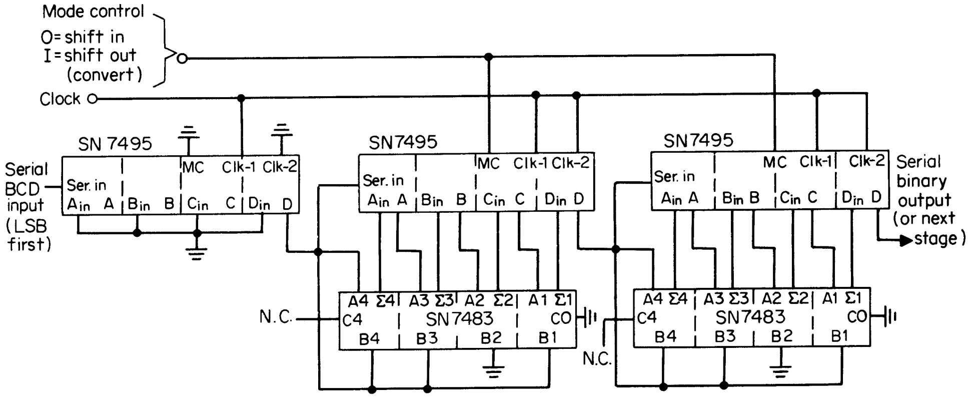

Mode control
0 = shift in
1 = shift out
(convert)
Clock
SN 7495
SN7495
SN7495
Serial BCD input (LSB first)
Ser. in
MC
Clk-1
Clk-2
Ain A
Bin B
Cin C
Din D
SN7483
A4 Σ4
A3 Σ3
A2 Σ2
A1 Σ1
C4
CO
B4
B3
B2
B1
N.C.
N.C.
Serial binary output (or next stage)

Fig. 9.42. Serial BCD to serial binary converter (three decades).

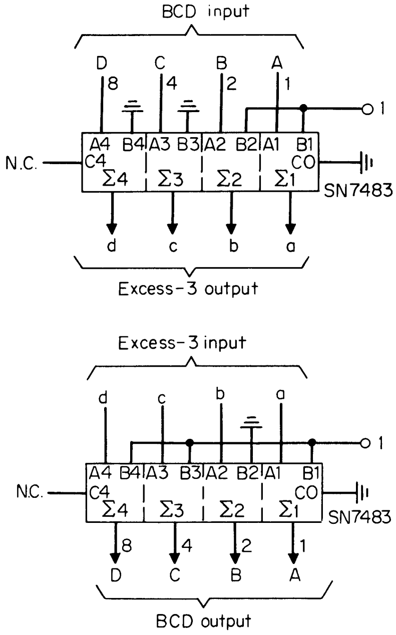

BCD input
D
C
B
A
8
4
2
1
1
A4 B4 A3 B3 A2 B2 A1 B1
N.C.
C4
CO
Σ4 Σ3 Σ2 Σ1
SN7483
d
c
b
a
Excess-3 output
Excess-3 input
d
c
b
a
1
A4 B4 A3 B3 A2 B2 A1 B1
N.C.
C4
CO
Σ4 Σ3 Σ2 Σ1
SN7483
8
4
2
1
D
C
B
A
BCD output

Fig. 9.43. BCD-to-excess-3 and excess-3-to-BCD conversion.

are first shifted into the SN7495 shift registers (mode control = 0); then the mode control is set to 1 for parallel input, and the converted binary number is shifted out. In the shifting-out process, the adders effectively subtract 3 when a 1 crosses a decade. This is accomplished by adding the 2's complement of 3 (1101) to the contents of the decade.

Conversion between BCD and Excess-3. Conversion between BCD and excess-3 is accomplished by the constant addition of 3 (see Fig. 9.43). Excess-3 to BCD is accomplished by the subtraction of 3 or, to state it more accurately, by the addition of the 2's complement of 3 (1101).

10

Counters

Flip-flops programmed as counters are found in almost every kind of digital equipment. They are used not only for counting but for equipment operation sequencing, frequency division, and mathematical manipulation as well.

In the most basic sense, counters are memory systems, in that they "remember" how many clock pulses have been applied to the input. The sequence in which information is stored is dependent upon application requirements and the discretion of the logic designer. Many of the more popular counters are available as off-the-shelf integrated circuits.

10.1 RIPPLE COUNTERS

The ripple counter is a basic counter commonly implemented with integrated circuits. Of all counters it is the simplest in logic and therefore easiest to design. The ripple counter is limited, however, in its speed of operation. Since the flip-flops in the ripple counter are not under command of a single clock pulse, it is an asynchronous counter.

Figure 10.1 shows a simple 4-bit binary ripple counter. Initially all flip-flops are in the logical 0 state ($Q_A = Q_B = Q_C = Q_D = 0$). A clock pulse is applied to the clock input of flip-flop A causing Q_A to change from logical 0 to logical 1. Flip-flop B does not change state since it is triggered by the negative-going edge of the clock pulse, i.e., by its clock input changing from logical 1 to logical 0. With the arrival of the second clock pulse to flip-flop A, Q_A goes from 1 to 0. This change of state creates the negative-going pulse edge needed to trigger flip-flop B, and thus Q_B goes from 0 to 1. Before arrival of the sixteenth clock pulse all flip-flops are in the 1 state. Clock pulse 16 causes Q_A, Q_B, Q_C, and Q_D to go to 0 in turn.

A 4-bit binary counter repeats itself for every 2^N ($N =$ number of flip-flops) clock pulses. This counter sequences in a number system of radix 16 and has 16 discrete states from 0 to $N - 1$. The 16 binary states are shown in Fig. 10.2.

In applications where the binary states of the counter must be converted into discrete outputs, a decoding network is provided. In Fig. 10.3 a 3-bit binary counter is shown with decoding of its eight possible states from 0 to 7.

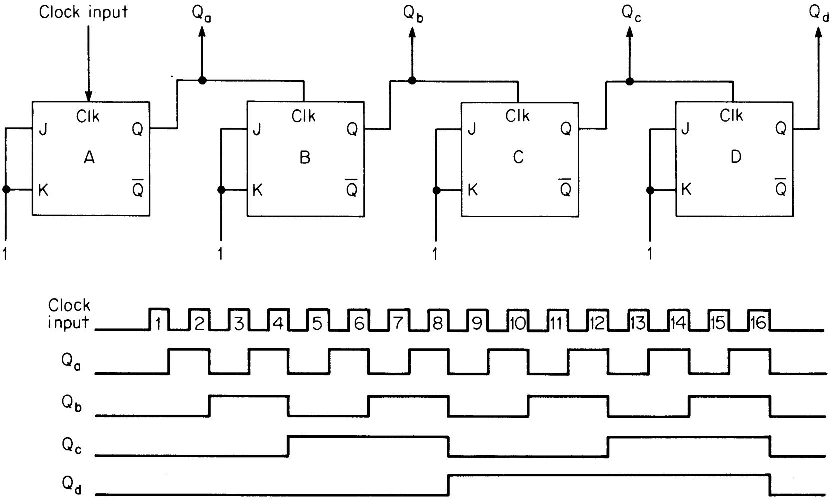


Fig. 10.1. 4-stage binary ripple counter.

In decoding the states of a ripple counter, spikes occur at decode matrix outputs as counter flip-flops change state. The propagation delay of the flip-flops creates these false states for only a short time, as shown in Fig. 10.4. Decode spikes are possible in any counter unless all flip-flops change state at exactly the same time or only one flip-flop changes state for any clock pulse. To eliminate the spikes at decode matrix outputs, a strobe pulse is used (see Fig. 10.3). The strobe pulse allows decoding to occur only after all flip-flops in the counter have become stable.

Maximum clock frequency for a counter is given by

$$\frac{1}{f} \leq N(T_p) + T_s$$

where N = number of flip-flop stages
T_p = propagation delay of one flip-flop
T_s = strobe time, width of decoded output pulse

Assuming each flip-flop in the counter shown in Fig. 10.1 has a propagation delay of 50 ns, 200 ns is then required for the counter to change from 1111 to 0000. And if decoding of any state requires 100 ns, then

$$\frac{1}{f} \geq 4(50) + 100 = 300 \text{ ns}$$

$$f \leq 3.67 \text{ MHz}$$

Flip-flop *A* in the counter of Fig. 10.3 changes state with each clock pulse and therefore divides the input clock frequency by 2. Flip-flop *B* changes state with every other clock pulse, dividing the frequency by 4. A 4-stage counter can be used to divide by 16 (2^n, n = number of flip-flops). Additional stages can be added if

State	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0

Fig. 10.2. State table, 4-bit ripple counter.

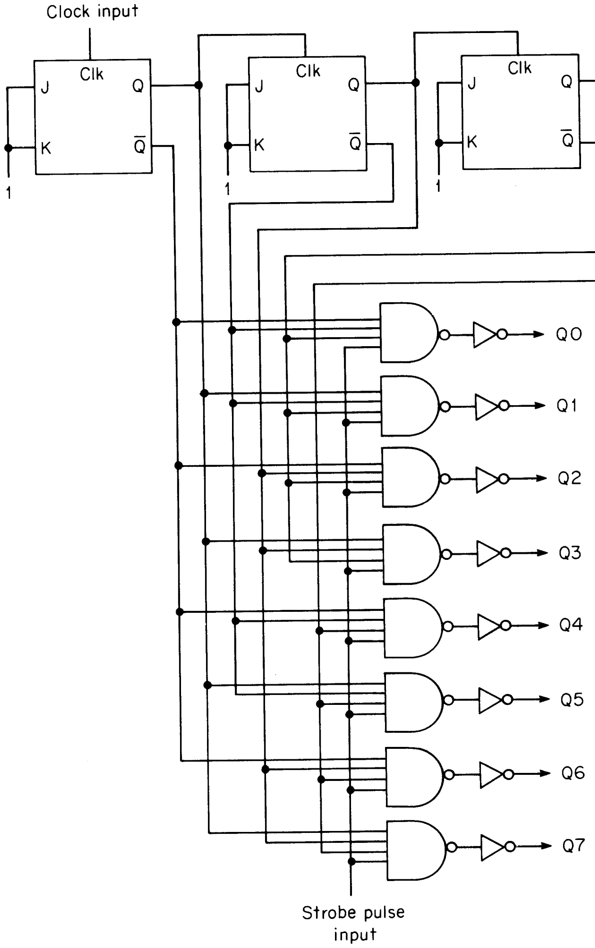
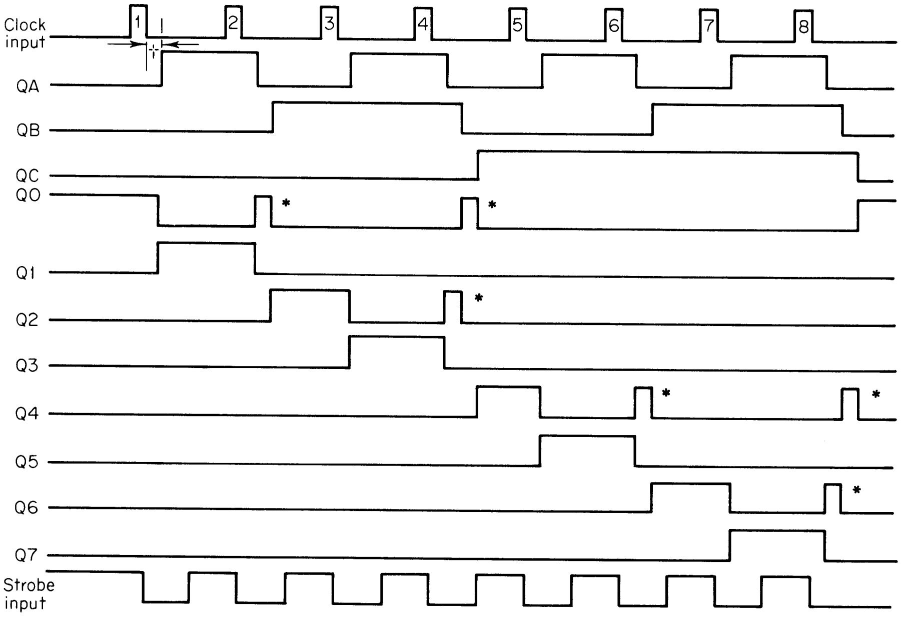

Clock input
Clk
J
Q
K
Q̄
1
Clk
J
Q
K
Q̄
1
Clk
J
Q
K
Q̄
1
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Strobe pulse input

Fig. 10.3. 3-bit ripple counter with decoded outputs.



* Decoding spikes † Typical propagation delay time (exaggerated scale)

Fig. 10.4. Timing diagram, 3-bit ripple counter.

division by some higher power of 2 is required. For division by any integer the following method can be used:

1. Find the number n of flip-flops required:

$$2^{n-1} \leq N \leq 2^n$$

where $N =$ counter cycle length. If N is not a power of 2, use the next higher power of 2.
2. Connect all flip-flops as a ripple counter (see Fig. 10.5).
3. Find the binary number $N - 1$.
4. Connect all flip-flop outputs that are 1 at the count $N - 1$ as inputs to a NAND gate. Also feed the clock pulse to the NAND gate.
5. Connect the NAND gate output to the preset inputs of all flip-flops for which $Q = 0$ at the count $N - 1$.

The counter resets in the following manner. At the positive-going edge of the Nth clock pulse all flip-flops are preset to the 1 state. On the trailing edge of the same clock pulse all flip-flops count to the 0 state; i.e., the counter recycles. For $N = 10$:

1. $2^3 \leq 10 \leq 2^4$. Thus, four flip-flops are required.
2. $N = 10$: 0101 (least significant bit, leftmost).
3. $N - 1 = 9$: 1001 (LSB, leftmost).
4. Connect as shown in Fig. 10.5.

Each flip-flop in a counter like the one in Fig. 10.5 has a specific decimal weight assigned to it. Flip-flop A has a weight of 2^0 (or 1) when its output is a logical 1. Flip-flop B has a weight of 2^1 (or 2), C has a weight of 2^2 (or 4), and D has a weight of 2^3 (or 8). The number stored in the counter at any specific time can be determined by summing decimal weights of flip-flops in the 1 state. A counter that counts in a standard binary manner and recycles for every 10 clock pulses is referred to as an 8-4-2-1 BCD (binary-coded decimal) counter.

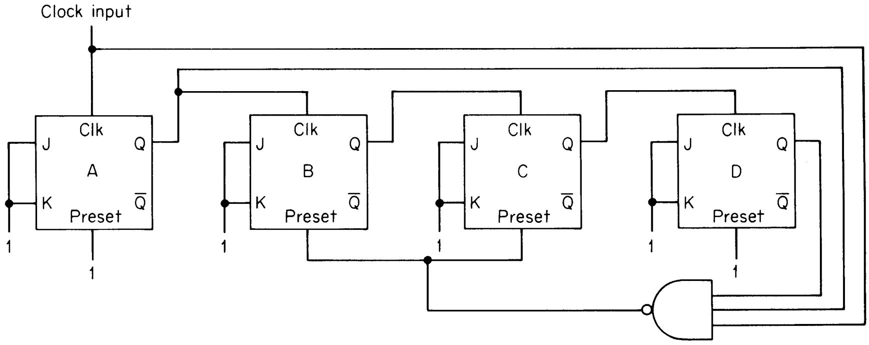
In many IC counter packages, the preset lines shown in Fig. 10.5 do not exist; only a common clear (or reset) line is available. Figure 10.6 shows a divide-by-12 counter using a common reset line. This counter is designed using the following procedure:

1. Find the number n of flip-flops required:

   $$2^{n-1} \le N \le 2^n$$

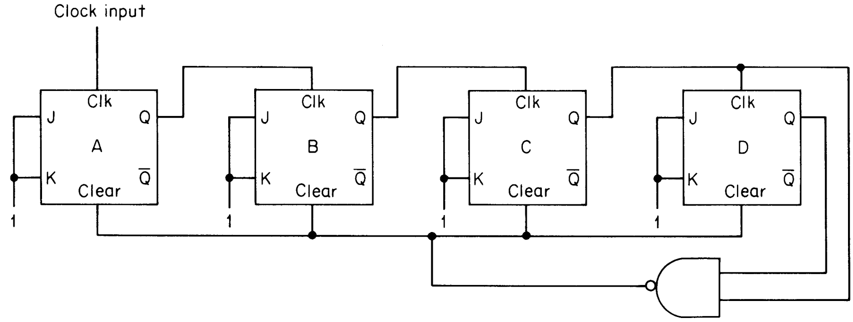
   where N = counter cycle length. If N is not a power of 2, use the next higher power of 2.
2. Connect all flip-flops as a ripple counter.
3. Find the binary number N.
4. Connect all flip-flop outputs for which $Q = 1$ at the count N, as inputs to a NAND gate. Connect the NAND gate output to the reset input of the counter.

When the counter reaches its Nth state, output of the NAND gate goes to a logical 0, resetting all flip-flops to 0. Although this is the simplest method for resetting



State	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
0	0	1/0	1/0	0

Fig. 10.5. BCD decade ripple counter.



State	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
0	1/0	1/0	0	0

Fig. 10.6. Divide-by-12 ripple counter.

ripple counters, and will be used extensively in this chapter, it is not considered the most reliable. If propagation delay from reset input to flip-flop output varies from stage to stage, the negative reset pulse may not be wide enough to reset all flip-flops to 0. For example, if one flip-flop resets in 10 ns and another resets in 50 ns, the reset pulse will exist for only 10 ns, and the slower flip-flop may not reset. Wide variation in reset propagation time is especially prevalent when counter outputs are unevenly loaded.

A good way to eliminate the problems encountered in resetting is to use a latch as shown in Fig. 10.7.

10.2 SYNCHRONOUS COUNTERS

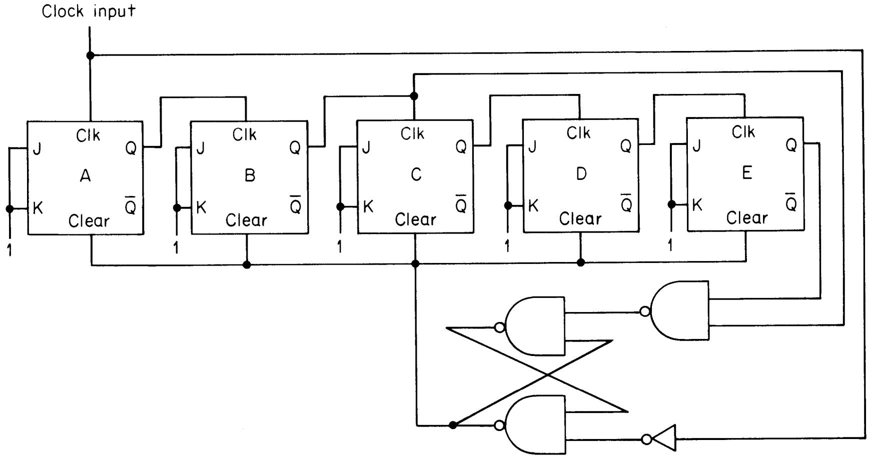
The synchronous counter eliminates the cumulative flip-flop delays seen in ripple counters. All flip-flops in a synchronous counter are under control of the same clock pulse. Repetition rate is limited only by the delay of any one flip-flop plus delays introduced by control gating. Design of synchronous counters for any number base other than some power of 2 is more difficult than design of a ripple counter, but the design is simplified through the use of the Karnaugh mapping technique.

Figure 10.8 shows a 4-bit synchronous counter with parallel carry. Parallel carry, also known as *carry look-ahead,* is the faster of the two methods of flip-flop control. According to the state table, flip-flop A is required to change state with occurrence of each clock pulse. Flip-flop B changes state when $Q_A = 1$. C changes state when $Q_A = Q_B = 1$, and D changes state when $Q_A = Q_B = Q_C = 1$. Control of flip-flop

A can be accomplished by tying J_A and K_A to logical 1. Control of flip-flop *B* is achieved by connecting J_B and K_B to Q_A. Control of flip-flop *C* can be achieved with the inverted output of a 2-input NAND gate whose inputs are Q_A and Q_B. Flip-flop *D* is controlled as *C* is, except that the NAND gate inputs are now Q_A, Q_B, and Q_C.

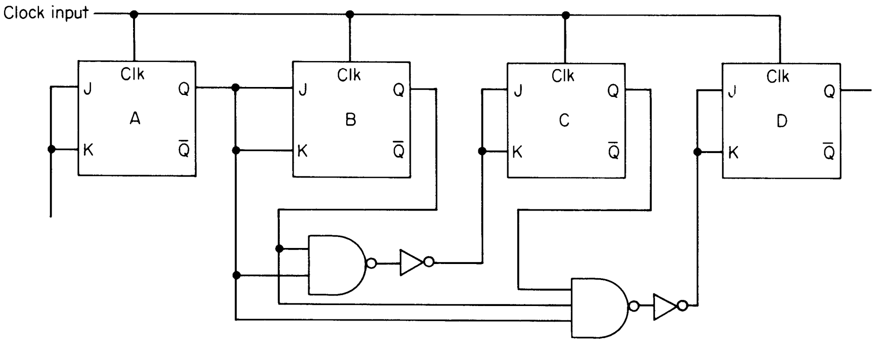
Synchronous counters for a binary count of cycle length 2^n can be designed once the control logic pattern is seen. For a cycle length that is not 2^n, control logic sometimes becomes quite confusing. It is for this reason that control matrices (Karnaugh maps) are drawn for each of the flip-flops. Figure 10.9 presets control matrices for the 4-bit synchronous counter of Fig. 10.8.

Figure 10.9*a* is an excitation table for a *J-K* flip-flop. Given the present state of a flip-flop, this table shows which input logic levels on the *J-K* inputs will produce the desired next flip-flop state. Figure 10.9*b* is the reference matrix for the state



State	Q_E	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0
19	1	0	0	1	1
0	1/0	0	1/0	0	0

Fig. 10.7. Divide-by-20 ripple counter using reset latch.



State	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0

Fig. 10.8. 4-stage synchronous counter with parallel carry.

assignments and shows how a 4-bit counter sequences through each of its 16 states.

Figure 10.9*c* is a control matrix for flip-flop *A*. Each square, which is also called a cell, represents one of 16 possible counter states. If the counter is at 0000 (LSB leftmost), the next state is 1000. To make flip-flop *A* change state, *J* must be a logical 1 and the logic level on *K* does not matter; hence cell 0000 shows 1*X*. The counter is now at 1000, and its next state will be 0100. To make flip-flop *A* change state again, *K* must now be 1 and the logical level on *J* does not matter; hence cell 1000 shows *X*1. This reasoning pattern is continued until the control matrix is completely filled.

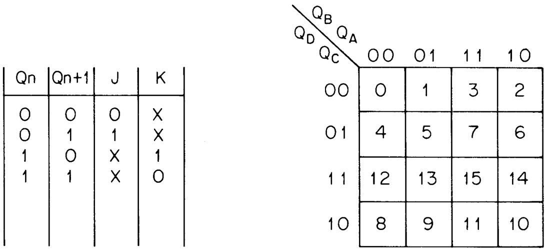
Figure 10.9*d* shows the control matrix for flip-flop *B*. When the counter is at 0000, the next state requires *B* to remain 0. $J = 0$, $K = X$ in cell 0000 will satisfy this requirement. In cell 1000, 1*X* is entered so that the next counter state will be 0100. For counter state 1100 to occur, 1*X* is placed in cell 0100.

When all control matrices are completed, each is examined and Boolean expressions for controlling flip-flops are determined. $J_A = K_A = 1$ becomes the control equation for flip-flop *A*. Control equations for the other three flip-flops are $J_B = K_B = Q_A$, $J_C = K_C = Q_A \cdot Q_B$, and $J_D = K_D = Q_A \cdot Q_B \cdot Q_C$. Since NAND gates perform the required Boolean functions, they are used for controlling flip-flops C and D.

Maximum clock frequency for a 4-bit synchronous counter with parallel carry is easily formulated:

$$\frac{1}{f} \geq T_p + T_g$$

where T_p = propagation delay of one flip-flop
T_g = propagation delay from input to output of control gating (in this case, the delay of one NAND gate and one inverter)

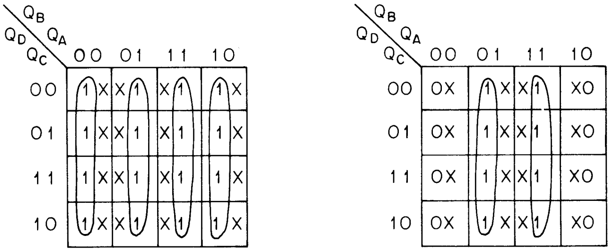


Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

X = "don't care"
(a) Excitation table

$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

(b) Reference matrix

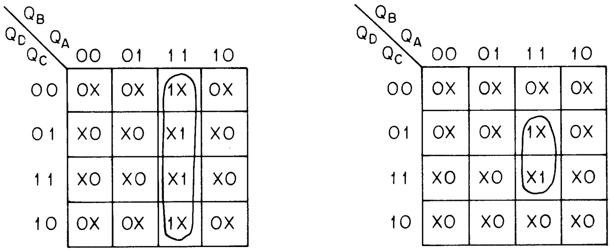


$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
00	1X	X1	X1	1X
01	1X	X1	X1	1X
11	1X	X1	X1	1X
10	1X	X1	X1	1X

$J_A = K_A = 1$
(c) Control matrix flip-flop A

$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
00	0X	1X	X1	X0
01	0X	1X	X1	X0
11	0X	1X	X1	X0
10	0X	1X	X1	X0

$J_B = K_B = Q_A$
(d) Control matrix flip-flop B



$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
00	0X	0X	1X	0X
01	X0	X0	X1	X0
11	X0	X0	X1	X0
10	0X	0X	1X	0X

$J_C = K_C = Q_A \cdot Q_B$
(e) Control matrix flip-flop C

$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
00	0X	0X	0X	0X
01	0X	0X	1X	0X
11	X0	X0	X1	X0
10	X0	X0	X0	X0

$J_D = K_D = Q_A \cdot Q_B \cdot Q_C$
(f) Control matrix flip-flop D

Fig. 10.9. Control matrices for 4-bit synchronous counter.

Legend for a K-map cell : JK

Assuming the propagation delay of any flip-flop to be 50 ns, and each control gate to have a delay of 25 ns,

$$\frac{1}{f} \geq 50 + 50 = 100 \text{ ns}$$

$$f \leq 10 \text{ MHz}$$

As the number of stages in a synchronous counter with parallel carry increases, the flip-flops must drive an ever-increasing number of NAND gates. Similarly, the number of inputs per control gate also increases. Ripple carry, shown in Fig. 10.10, eliminates these difficulties, but the clock speed of the counter is reduced. Reduction of clock speed stems from the fact that the delay through control logic is now 100 ns instead of the 50 ns achieved with parallel carry. Maximum clock frequency for ripple carry is given by

$$\frac{1}{f} \geq T_p + (N - 2)T_g$$

where N = number of flip-flop stages
T_p = propagation delay of one flip-flop
T_g = propagation delay of one stage of control gating (in this case the delay of one NAND gate and one inverter)

$$\frac{1}{f} \geq 50 + 2(50) = 150 \text{ ns}$$

$$f \leq 6.67 \text{ MHz}$$

Cycle length of a synchronous counter is defined by 2^N (N = number of flip-flops). An 8-4-2-1 BCD decade counter is the most common counter with a cycle length differing from 2^N. Figure 10.11 shows control matrices for this counter. Note that states 10 through 15 are not used in the BCD count and are therefore eliminated from control matrices. From the derived control equations a counter as shown in Fig. 10.12 is designed.

In synchronous counters a strobe pulse is not usually required when decoding the counter states. Falsely decoded outputs may occur when flip-flop propagation

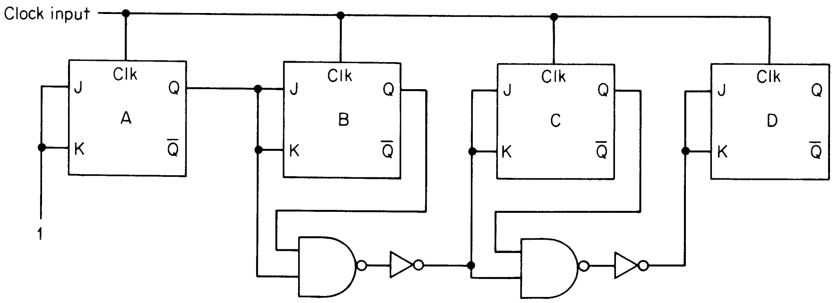
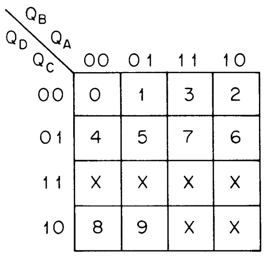
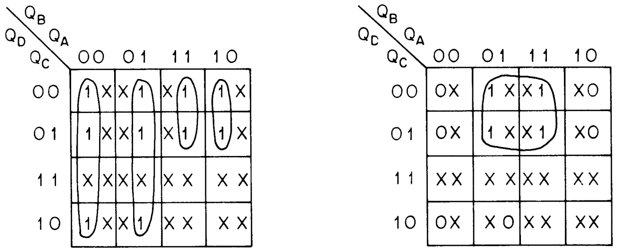


Fig. 10.10. 4-stage synchronous counter with ripple carry.



	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X	X	X	X
10	8	9	X	X

(a) Reference matrix

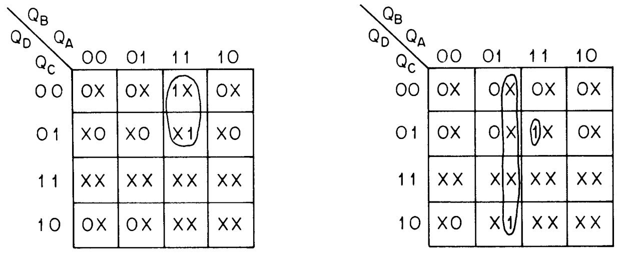


	00	01	11	10
00	1X	X1	X1	1X
01	1X	X1	X1	1X
11	XX	XX	XX	XX
10	1X	X1	XX	XX

$J_A = K_A = 1$
(b) Control matrix flip-flop A

	00	01	11	10
00	0X	1X	X1	X0
01	0X	1X	X1	X0
11	XX	XX	XX	XX
10	0X	X0	XX	XX

$J_B = K_B = Q_A \cdot \overline{Q}_D$
(c) Control matrix flip-flop B



	00	01	11	10
00	0X	0X	1X	0X
01	X0	X0	X1	X0
11	XX	XX	XX	XX
10	0X	0X	XX	XX

$J_C = K_C = Q_A \cdot Q_B$
(d) Control matrix flip-flop C

	00	01	11	10
00	0X	0X	0X	0X
01	0X	0X	1X	0X
11	XX	XX	XX	XX
10	X0	X1	XX	XX

$J_D = Q_A \cdot Q_B \cdot Q_C$ $K_D = Q_A$
(e) Control matrix flip-flop D

Legend for a K-map cell | JK |

Fig. 10.11. Control matrices for a decade counter.

delays between stages vary. Maximum width of any false output will not exceed the propagation delay time difference between the slowest and the fastest flip-flop.

For some applications, the counter must be capable not only of counting up, but also of counting down. Figure 10.13 shows control matrices for a 4-stage binary up/down counter. Each matrix has one variable more than an up-only counter. For $X = 1$ the counter sequences in a standard binary code. For $X = 0$ it counts down (0000, 1111, 0111, 1011, . . . , 1000, 0000, LSB leftmost). The derived control equations are implemented in Fig. 10.14 for both parallel and serial carry.

Control equations for up/down counters with cycle lengths not equal to 2^N can be found by a technique similar to what was shown in Fig. 10.11. All that is required

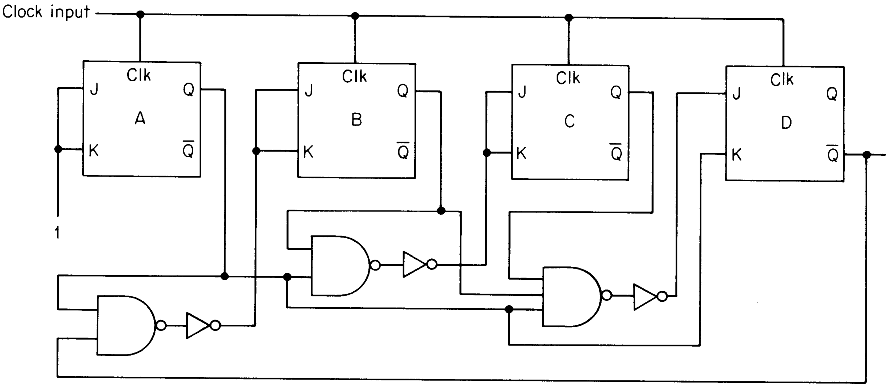


Fig. 10.12. Synchronous decade counter with parallel carry.

is to ignore unused states and force the counter to return to 0000 after the $N - 1$ state.

10.3 SERIES 54/74 COUNTERS

Counters are available in Series 54/74 to fill many of the usual requirements. Table 10.1 is a summary table covering these counters. The first four listed (SN54/7490 through SN54/74L93) are ripple counters. The remaining four are synchronous.

SN54/7490 Decade Counter. This decade counter (Fig. 10.15) consists of four master-slave flip-flops internally interconnected to provide a divide-by-2 counter and a divide-by-5 counter. Gated direct-reset lines are provided to inhibit count inputs and return all outputs to a logical 0 or to a binary-coded-decimal (BCD) count of 9. Since the output from flip-flop *A* is not internally connected to the succeeding stages, the count may be separated into three independent count modes:

1. When used as a binary-coded-decimal decade counter, the *B* input must be externally connected to the *A* output. The *A* input receives the incoming count, and a count sequence is obtained in accordance with the BCD count sequence truth table (Fig. 10.15). In addition to a conventional 0 reset, inputs are provided to reset a BCD count of 9 for 9's complement decimal applications.
2. If a symmetrical divide-by-10 count is desired for frequency synthesizers or other applications requiring division of a binary count by a power of 10, the *D* output must be externally connected to the *A* input. The input count is then applied at the *B* input, and a divide-by-10 square wave is obtained at output *A*.
3. For operation as a divide-by-2 counter and a divide-by-5 counter, no external interconnections are required. Flip-flop *A* is used as a binary element for the divide-by-2 function. The *B* input is used to obtain binary divide-by-5 operation at the *B*, *C*, and *D* outputs. In this mode, the two counters operate independently; however, all four flip-flops are reset simultaneously.

X	$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
X = 1	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10
X = 0	10	8	7	5	6
	11	4	3	1	2
	01	12	11	9	10
	00	0	15	13	14

(a) Reference matrix

X	$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
X = 1	00	1X	X1	X1	1X
	01	1X	X1	X1	1X
	11	1X	X1	X1	1X
	10	1X	X1	X1	1X
X = 0	10	1X	X1	X1	1X
	11	1X	X1	X1	1X
	01	1X	X1	X1	1X
	00	1X	X1	X1	1X

$J_A = K_A = 1$

(b) Control matrix flip-flop A

Legend : for a K-map cell $\boxed{J \quad K}$

X	$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
X = 1	00	0X	1X	X1	X0
	01	0X	1X	X1	X0
	11	0X	1X	X1	X0
	10	0X	1X	X1	X0
X = 0	10	1X	0X	X0	X1
	11	1X	0X	X0	X1
	01	1X	0X	X0	X1
	00	1X	0X	X0	X1

$J_B = K_B = X \cdot Q_A + \overline{X}\,\overline{Q}_A$

(c) Control matrix flip-flop B

X	$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
X = 1	00	0X	0X	1X	0X
	01	X0	X0	X1	X0
	11	X0	X0	X1	X0
	10	0X	0X	1X	0X
X = 0	10	1X	0X	0X	0X
	11	X1	X0	X0	X0
	01	X1	X0	X0	X0
	00	1X	0X	0X	0X

$J_C = K_C = X \cdot Q_A \cdot Q_B + \overline{X} \cdot \overline{Q}_A \overline{Q}_B$

(d) Control matrix flip-flop C

X	$Q_DQ_C \backslash Q_BQ_A$	00	01	11	10
X = 1	00	0X	0X	0X	0X
	01	0X	0X	1X	0X
	11	X0	X0	X1	X0
	10	X0	X0	X0	X0
X = 0	10	X1	X0	X0	X0
	11	X0	X0	X0	X0
	01	0X	0X	0X	0X
	00	1X	0X	0X	0X

$J_D = K_D = X \cdot Q_A \cdot Q_B \cdot Q_C + \overline{X} \cdot \overline{Q}_A \cdot \overline{Q}_B \cdot \overline{Q}_C$

(e) Control matrix flip-flop D

Fig. 10.13. Control matrices for synchronous binary up/down counter.

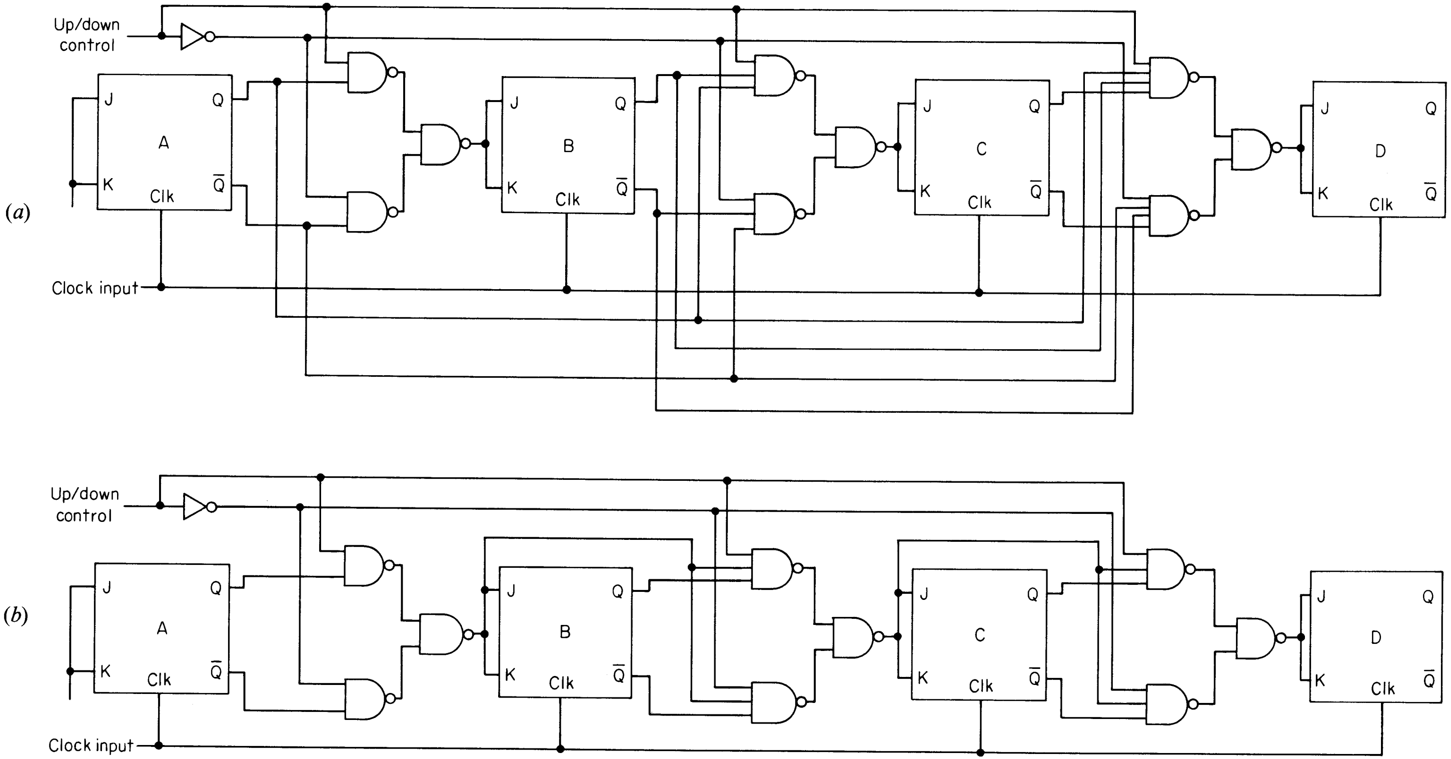
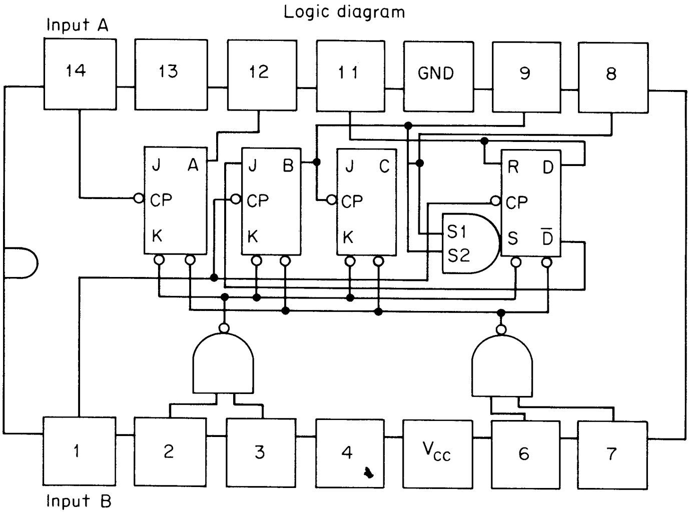

(a)
Up/down control
Clock input
A
B
C
D
J
K
Q
$\bar{Q}$
Clk
(b)
Up/down control
Clock input

Fig. 10.14. 4-stage up/down binary counters with (*a*) parallel carry, (*b*) ripple carry.

Table 10.1. **Summary of Series 54/74 Counters**

Types	Mode	Fully programmable	Clear	Bidirectional	Over-/under-flow output	Typical clock frequency, max	Typical power dissipation, mW
SN54/7490	÷2, ÷5, ÷10 Decade	No, preset to 1001 only	Gated	No	No	18 MHz	160
SN54/7492	÷2, ÷3, ÷6, ÷12	No	Gated	No	No	18 MHz	155
SN54/7493 and SN54/74L93	÷2, ÷4, ÷8, ÷16 Binary	No	Gated	No	No	18 MHz 3 MHz	160 16
SN54/74190	Synchronous Decade	Direct	No	Yes	Yes	30 MHz	325
SN54/74191	Synchronous Binary	Direct	No	Yes	Yes	30 MHz	325
SN54/74192	Synchronous Decade	Direct	Yes	Yes	Yes	32 MHz	325
SN54/74193	Synchronous Binary	Direct	Yes	Yes	Yes	32 MHz	325



Propagation delay 60 ns
Power dissipation 160 mW

Truth table

Mode 1 (BCD)				Mode 2 (symmetrical divide-by-10)				Mode 3 (divide-by-5)		
A	B	C	D	A	B	C	D	B	C	D
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	1	0	0
0	1	0	0	0	0	1	0	0	1	0
1	1	0	0	0	1	1	0	1	1	0
0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	1	0	0	0			
0	1	1	0	1	1	0	0			
1	1	1	0	1	0	1	0			
0	0	0	1	1	1	1	0			
1	0	0	1	1	0	0	1			

Figure 10.15

SN54/7492 Divide-by-12 Counter. This 4-bit binary counter consists of four master-slave flip-flops which are internally interconnected to provide a divide-by-2 counter and a divide-by-6 counter (see Fig. 10.16). A gated direct-reset line can inhibit the count inputs and simultaneously return the four flip-flop outputs to logical 0. Since the output from flip-flop *A* is not internally connected to the succeeding flip-flops, the counter may be operated in three modes:

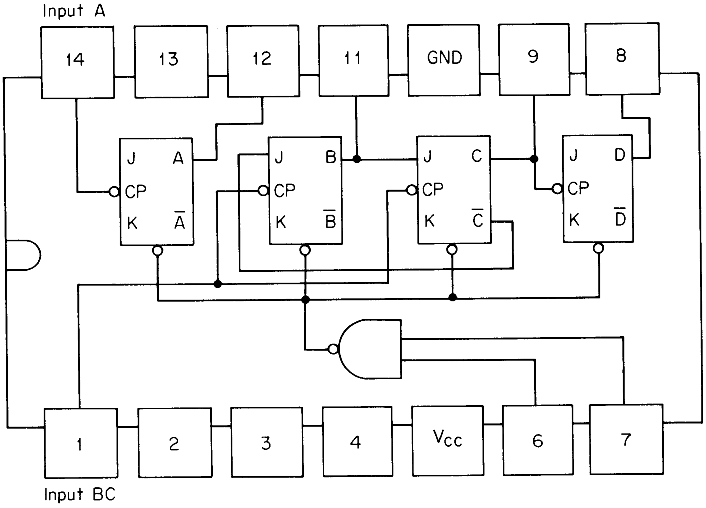
1. When used as a divide-by-12 counter, output *A* may be externally connected to input *BC*. Then input count pulses are applied to input *A*. Simultaneous divisions of 2, 6, and 12 are performed at the *A*, *C*, and *D* outputs as shown in the mode 1 truth table (Fig. 10.16).
2. When used as a divide-by-6 counter, the input count pulses are applied to input *BC*. Simultaneous frequency divisions of 3 and 6 are available at the

C and *D* outputs. Independent use of flip-flop *A* is available if the reset function coincides with reset of the divide-by-6 counter.

3. Another divide-by-12 code is available if output *D* is externally connected to input *A*. The input count pulses are then applied to input *BC*. Simultaneous frequency divisions of 3, 6, and 12 are available at the *C*, *D*, and *A* outputs.

SN54/7493 and SN54/74L93 Binary Counter. This 4-bit binary counter consists of four master-slave flip-flops which are internally interconnected to provide a divide-by-2 counter and a divide-by-8 counter (Fig. 10.17). A gated direct-reset line returns the four flip-flop outputs to logical 0. Since the output from flip-flop

Logic diagram



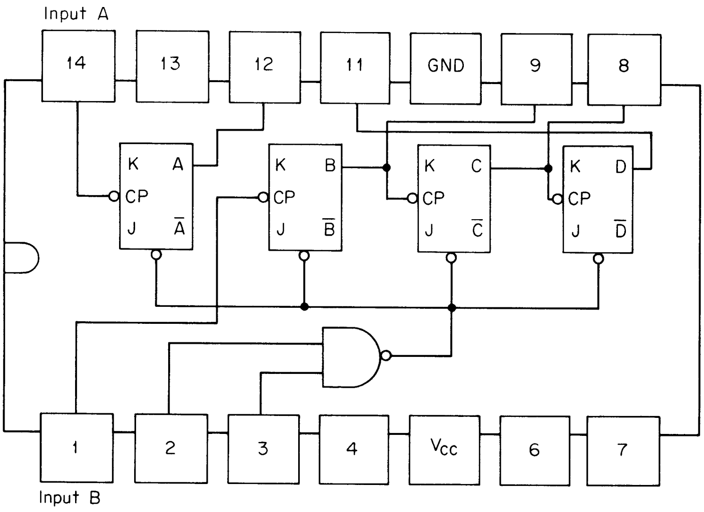
Propagation delay 60 ns
Power dissipation 160 mW

Truth table

Mode 1 (divide-by-12)				Mode 2 (divide-by-6)			Mode 3 (divide-by-12)			
A	B	C	D	B	C	D	A	B	C	D
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0	0	1	0
1	1	0	0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0	1	0	1
1	0	1	0	0	1	1	0	0	1	1
0	0	0	1				1	0	0	0
1	0	0	1				1	1	0	0
0	1	0	1				1	0	1	0
1	1	0	1				1	0	0	1
0	0	1	1				1	1	0	1
1	0	1	1				1	0	1	1

Figure 10.16

Logic diagram



Propagation delay 60 ns
Power dissipation 160 mW

Truth table

Mode 1 (divide-by-16)				Mode 2 (divide-by-8)		
D	C	B	A	B	C	D
0	0	0	0	0	0	0
0	0	0	1	1	0	0
0	0	1	0	0	1	0
0	0	1	1	1	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	0	1	1
0	1	1	1	1	1	1
1	0	0	0			
1	0	0	1			
1	0	1	0			
1	0	1	1			
1	1	0	0			
1	1	0	1			
1	1	1	0			
1	1	1	1			

Figure 10.17

A is not internally connected to the succeeding flip-flops, the counter may be operated in two independent modes:

1. When used as a 4-bit ripple-through counter, output *A* must be externally connected to input *B*. The input count pulses are applied to input *A*. Simultaneous divisions of 2, 4, 8, and 16 are performed at the *A*, *B*, *C*, and *D* outputs as shown in the truth table (Fig. 10.17).
2. When used as a 3-bit ripple-through counter, the input count pulses are applied to input *B*. Simultaneous frequency divisions of 2, 4, and 8 are available at the *B*, *C*, and *D* outputs. Independent use of flip-flop *A* is

available if the reset function coincides with reset of the 3-bit ripple-through counter.

SN54/74190 and SN54/74191 Synchronous Up/Down Counters. SN54/74190 and SN54/74191 are synchronous 4-bit binary up/down counters with parallel carry. The former has modified steering logic for an 8-4-2-1 BCD decade count whereas the latter is connected for normal binary counting. Logic and timing diagrams for these devices are shown in Figs. 10.18 and 10.19.

Four master-slave flip-flops change state with a logical 0 to logical 1 transition of the count input. Direction of counting is determined by the state of the up/down control line. Logical 0 is count up, logical 1 is count down. The enable input allows the counter to be inhibited although clock pulses are present. A logical 0 enables the counter.

Both counters can be parallel-loaded at data inputs when the load is at 0. The load and clear operations are independent of clock and counter state. The SN54/74190 and SN54/74191 have been designed so that external logic is minimized when cascading packages. Two outputs are provided for this purpose. Maximum or minimum output is 1 if the counter either contains 15 and is counting up or contains 0 and is counting down. In the case of SN74190, maximum count is 9. Ripple count enable is 0 when enable input is 0, max or min output is a 1, and count input is 0.

These counters may be cascaded in three different ways, as illustrated by Figs. 10.20, 10.21, and 10.22.

Figure 10.20 shows packages connected as an asynchronous counter. Each IC package is synchronous within itself, but between stages the system is a ripple counter. The up/down control line must not be changed when the clock input is 0, nor should the up/down control be changed until the counter is in a stable state. Since ripple count enable is generated by gating action of the inverted count pulse and maximum output, the count pulse should be kept wide enough to strobe out any false ripple count enable outputs.

Figure 10.21 shows SN74190 and SN74191 used as a synchronous counter with ripple carry between integrated circuits. The count pulse must propagate through the steering logic of each IC reducing the maximum count frequency for each additional IC that is cascaded. Here again, the up/down control line should not be changed when the clock is 0.

A synchronous counter with parallel carry is shown in Fig. 10.22. Parallel carry allows more stages to be cascaded without significant reduction of counting frequency. The only restriction on the number of stages that may be cascaded in this manner is due to loading of the max or min output by the external gating.

The SN54/74190 and SN54/74191 may also be used as programmable dividers. By presetting any number into the counter and counting to the maximum or minimum count, division is achieved. A schematic for this mode of operation is shown in Fig. 10.23. Upon receipt of count pulses, the counter sequences to maximum count. When the count input goes to a 0, the ripple count enable, which is connected to the load input, loads the number on the data inputs into the counter. The counting sequence then repeats.

Logic diagram

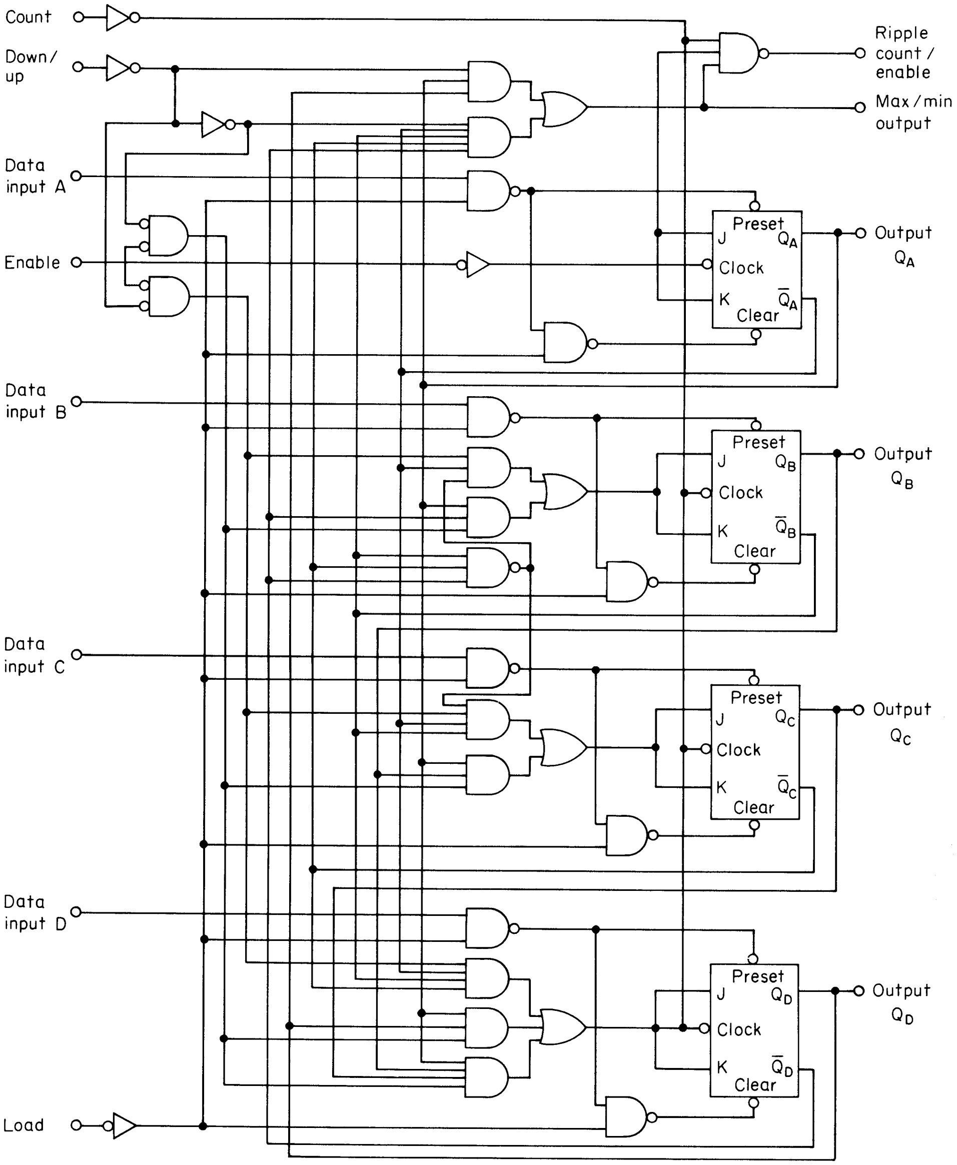

Count
Down/ up
Data input A
Enable
Data input B
Data input C
Data input D
Load
Ripple count / enable
Max/min output
Preset
J
Q_A
Clock
K
$\overline{Q}_A$
Clear
Output Q_A
Q_B
$\overline{Q}_B$
Output Q_B
Q_C
$\overline{Q}_C$
Output Q_C
Q_D
$\overline{Q}_D$
Output Q_D

Figure 10.18

Illustrated below is the following sequence :

1. Load (preset) to BCD seven
2. Count up to eight, nine (maximum), zero, one and two
3. Inhibit
4. Count down to one, zero (minimum), nine, eight and seven

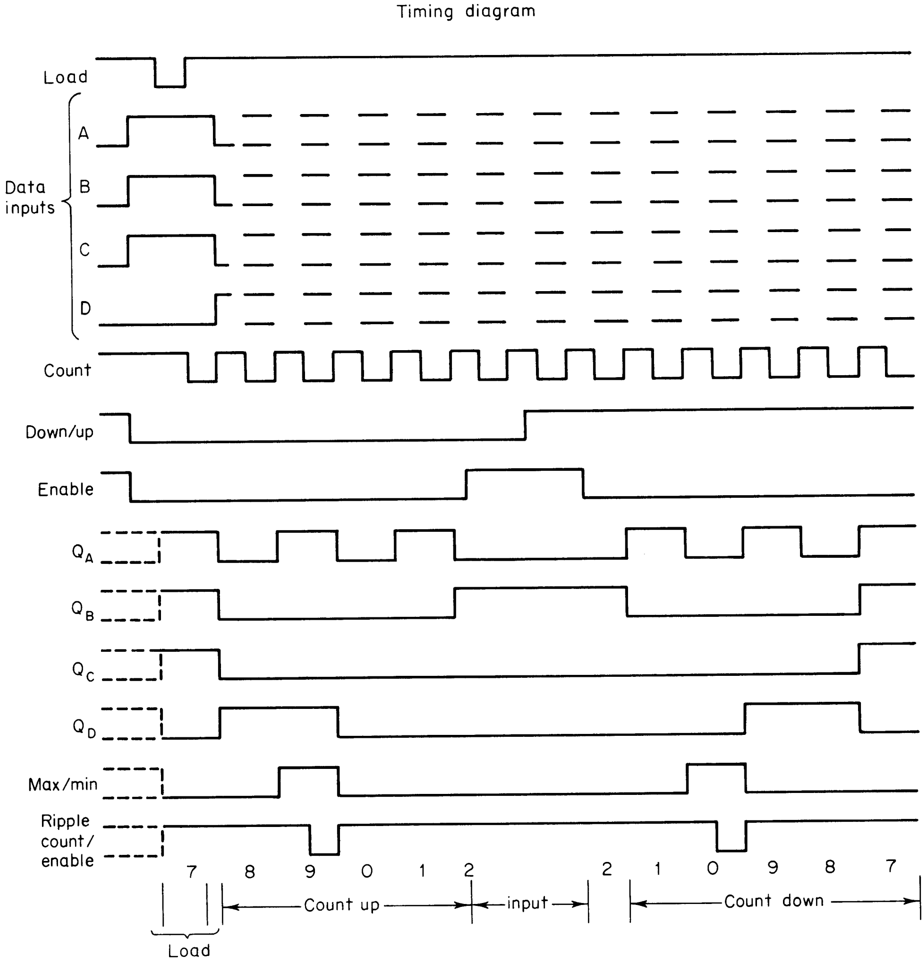

Timing diagram
Load
Data inputs
A
B
C
D
Count
Down/up
Enable
Q_A
Q_B
Q_C
Q_D
Max/min
Ripple count/ enable
7 8 9 0 1 2 2 1 0 9 8 7
Count up
input
Count down
Load

Figure 10.18. (*Continued*)

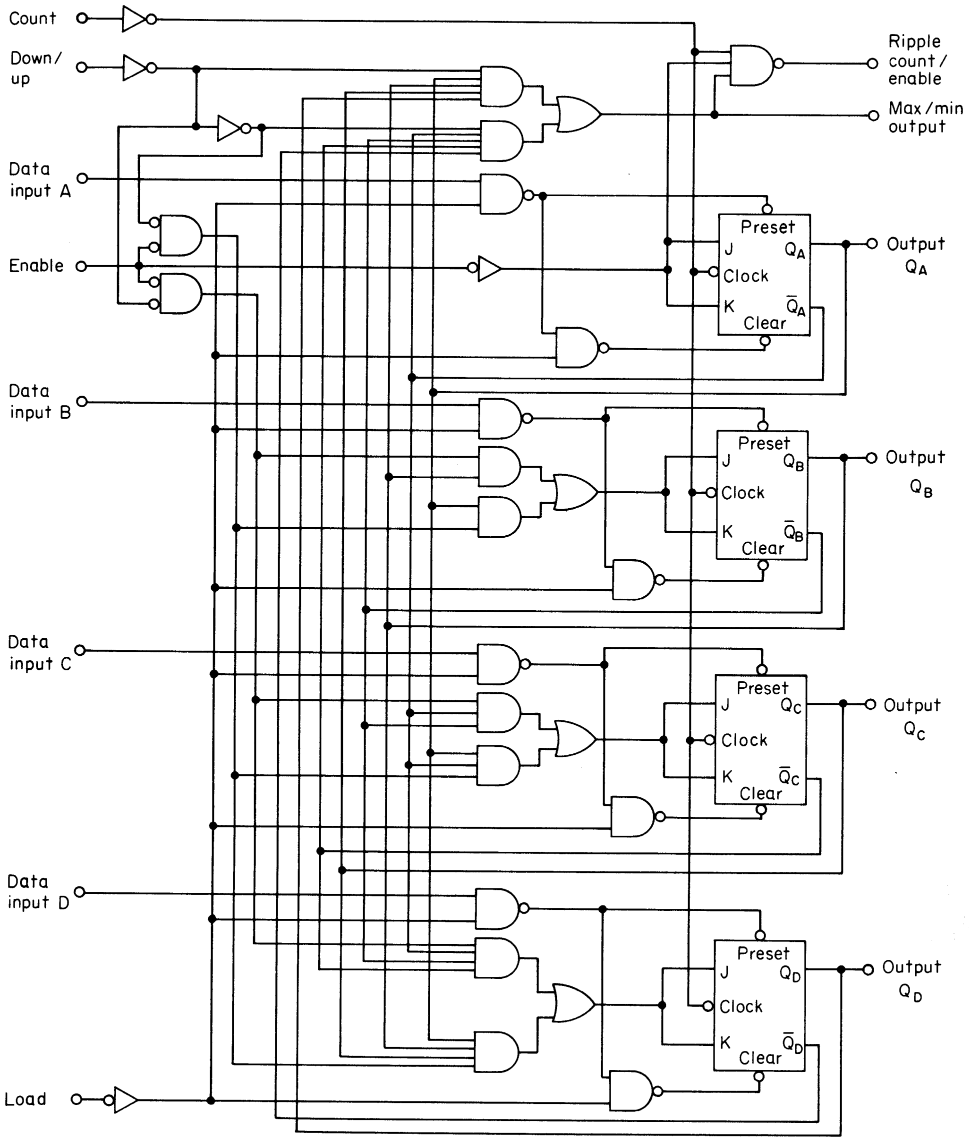

Count
Down/up
Data input A
Enable
Data input B
Data input C
Data input D
Load
Ripple count/enable
Max/min output
Preset
J
Clock
K
Clear
Q_A
$\overline{Q}_A$
Output Q_A
Q_B
$\overline{Q}_B$
Output Q_B
Q_C
$\overline{Q}_C$
Output Q_C
Q_D
$\overline{Q}_D$
Output Q_D

Figure 10.19

Typical load, count and inhibit sequences

Illustrated below is the following sequence:

1. Load (preset) to binary thirteen
2. Count up to fourteen, fifteen (maximum), zero, one and two
3. Inhibit
4. Count down to one, zero (minimum), fifteen, fourteen, and thirteen

Timing diagram

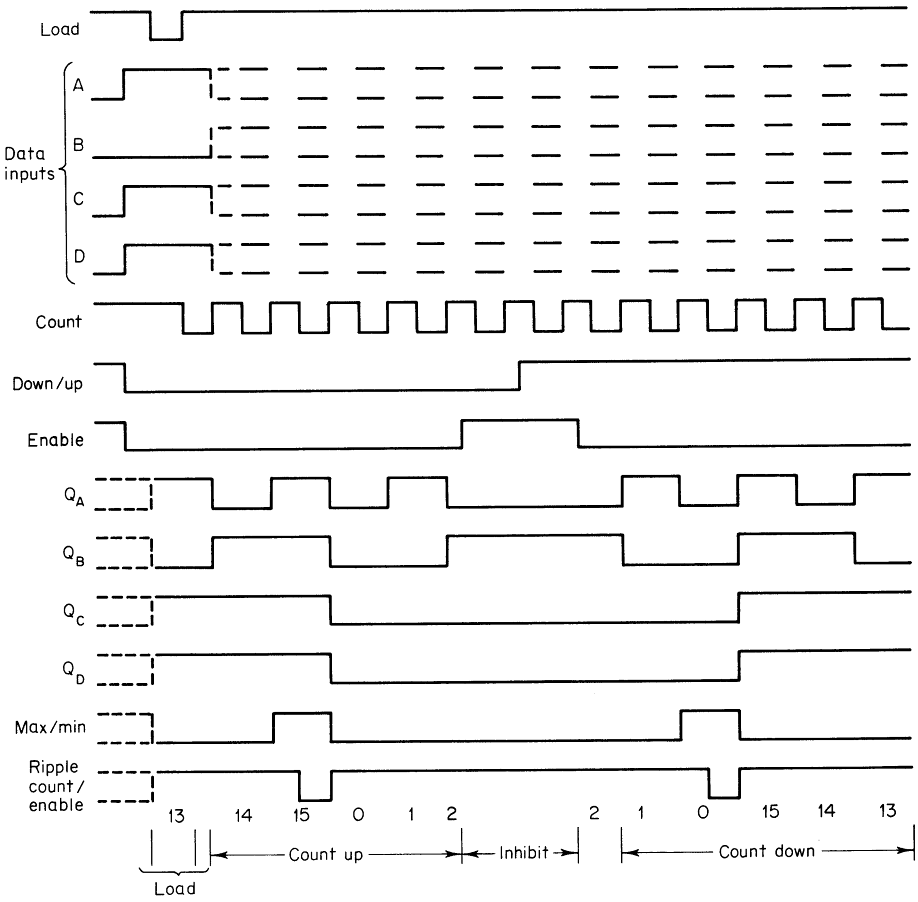

Load
Data inputs
A
B
C
D
Count
Down/up
Enable
Q_A
Q_B
Q_C
Q_D
Max/min
Ripple count/enable
13
14
15
0
1
2
2
1
0
15
14
13
Count up
Inhibit
Count down
Load

Figure 10.19. (*Continued*)

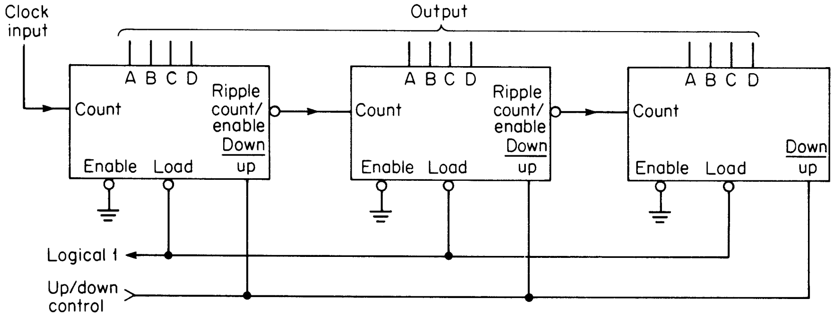

Clock input
Output
A B C D
Ripple count/ enable
Count
Down up
Enable
Load
Logical 1
Up/down control

Fig. 10.20. Ripple counter with cascaded SN54/74190 or SN54/74191 synchronous counters.

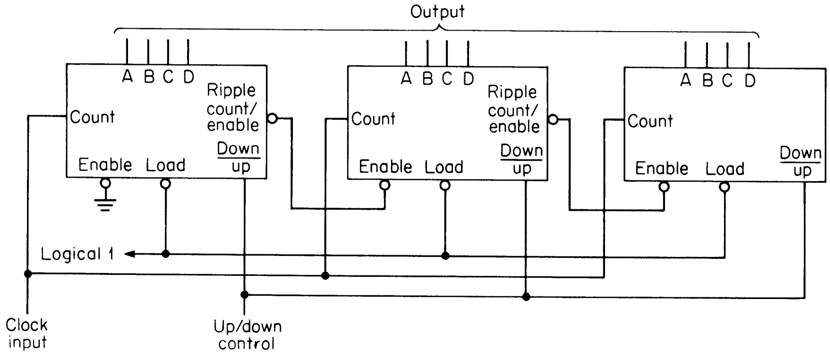

Output
A B C D
Ripple count/ enable
Count
Down up
Enable
Load
Logical 1
Clock input
Up/down control

Fig. 10.21. Synchronous counter with ripple carry with cascaded SN54/74190 or SN54/74191 synchronous counters.

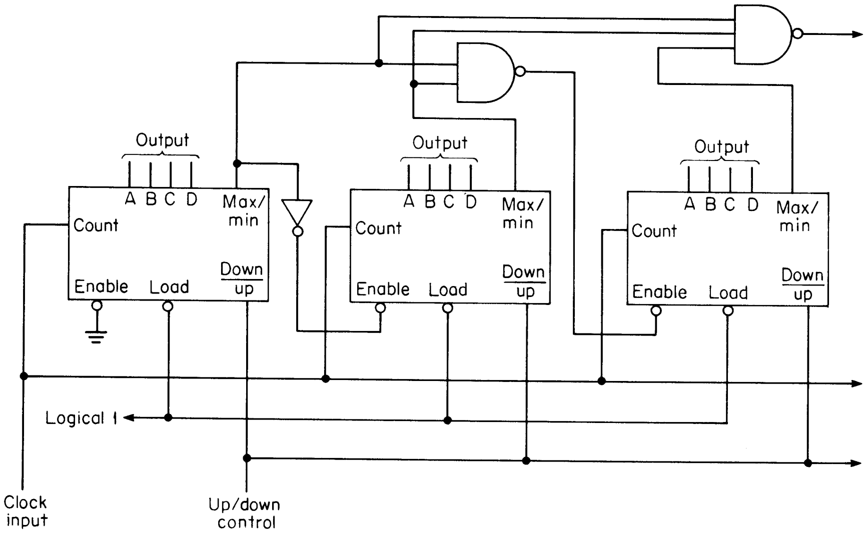
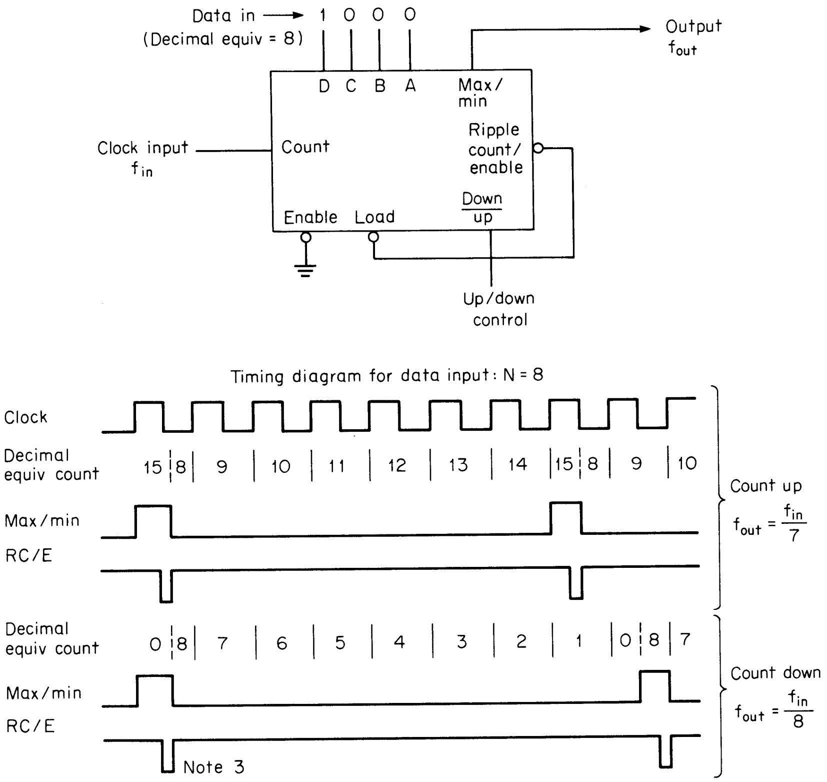

Output
A B C D
Max/ min
Count
Down up
Enable
Load
Logical 1
Clock input
Up/down control

Fig. 10.22. Synchronous counter with parallel carry with cascaded SN54/74190 or SN54/74191 synchronous counters.


Data in → 1 0 0 0
(Decimal equiv = 8)
D C B A
Max/min
Output
f_{out}
Clock input
f_{in}
Count
Ripple count/enable
Down/up
Enable Load
Up/down control
Timing diagram for data input: N = 8
Clock
Decimal equiv count
15 8 9 10 11 12 13 14 15 8 9 10
Max/min
RC/E
Count up
$f_{out} = \frac{f_{in}}{7}$
Decimal equiv count
0 8 7 6 5 4 3 2 1 0 8 7
Max/min
RC/E
Note 3
Count down
$f_{out} = \frac{f_{in}}{8}$

Notes: 1. For a down count: $f_{out} = \frac{f_{in}}{N}$ for $1 \leqslant N \leqslant 15$, where N is the data input

2. For an up count: $f_{out} = \frac{f_{in}}{15-N}$ for $0 \leqslant N \leqslant 14$, where N is the data input

3. The RC/E (ripple count/enable) pulse is typically 30 ns wide

Fig. 10.23. Divide-by-*N* counter using SN54/74191 synchronous binary counter.

SN54/74192 and SN54/74193 Synchronous Up/Down Counters. The SN54/74192 and SN54/74193 are synchronous 4-bit binary counters with parallel carry. The SN54/74193 is connected for normal binary counting, whereas the SN54/74192 has modified steering logic for an 8-4-2-1 BCD decade count. Logic and timing diagrams for these devices are shown in Figs. 10.24 and 10.25.

Four master-slave flip-flops change state with a 0-to-1 transition of count input. Direction of counting is a function of which count input is used. The unused clock input must be at a logical 1.

Both counters can be parallel-loaded at data inputs. When the load is at a logical 0, the load and clear operations are independent of clock and counter state. A 1 on the clear input resets all flip-flops to 0. The SN54/74192 and SN54/74193 have been designed so that external logic is minimized when cascading packages. Carry is 1 when the maximum count has been reached and count-up input is 0. Borrow-out is 1 when 0 has been reached and count-down input is 0.

Logic diagram

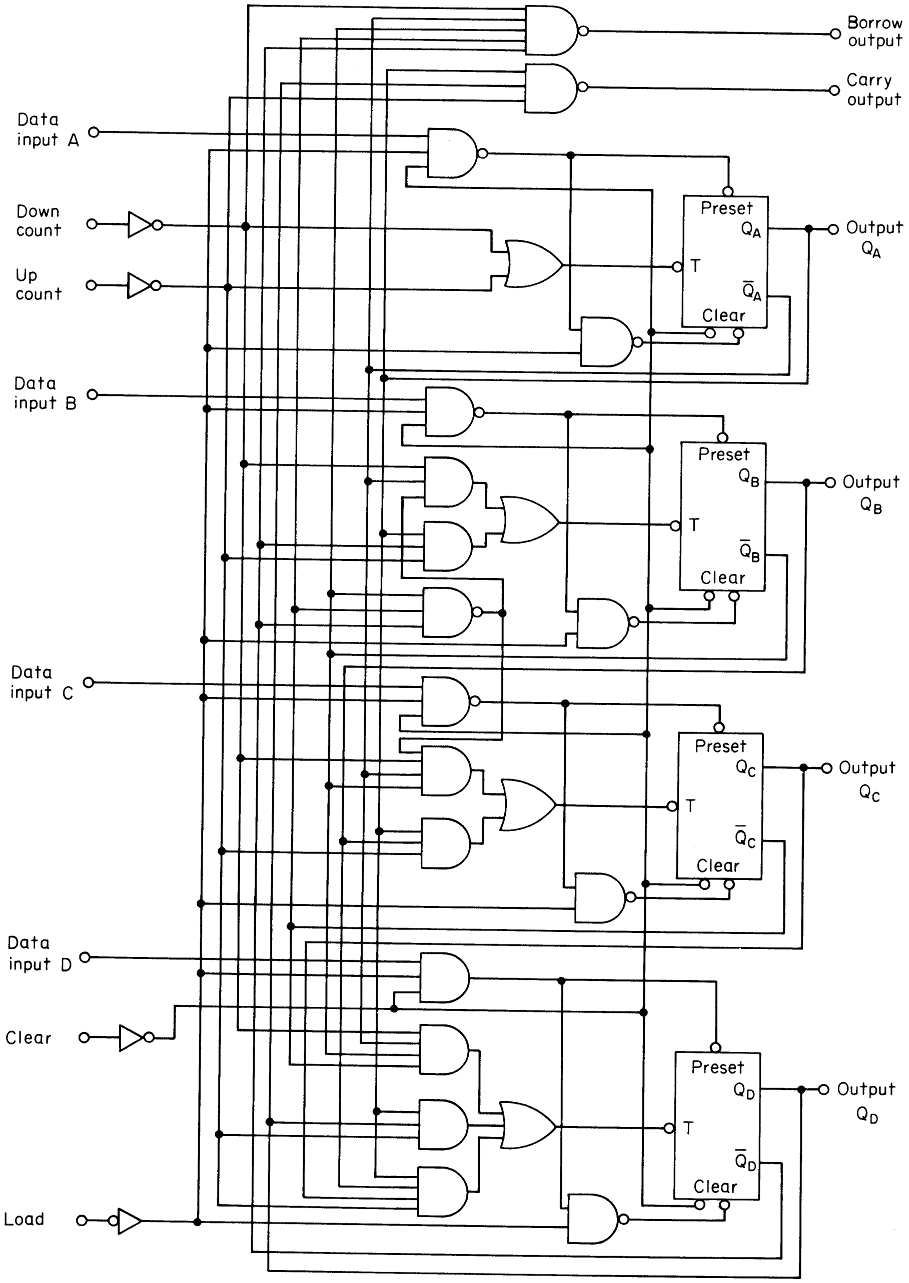
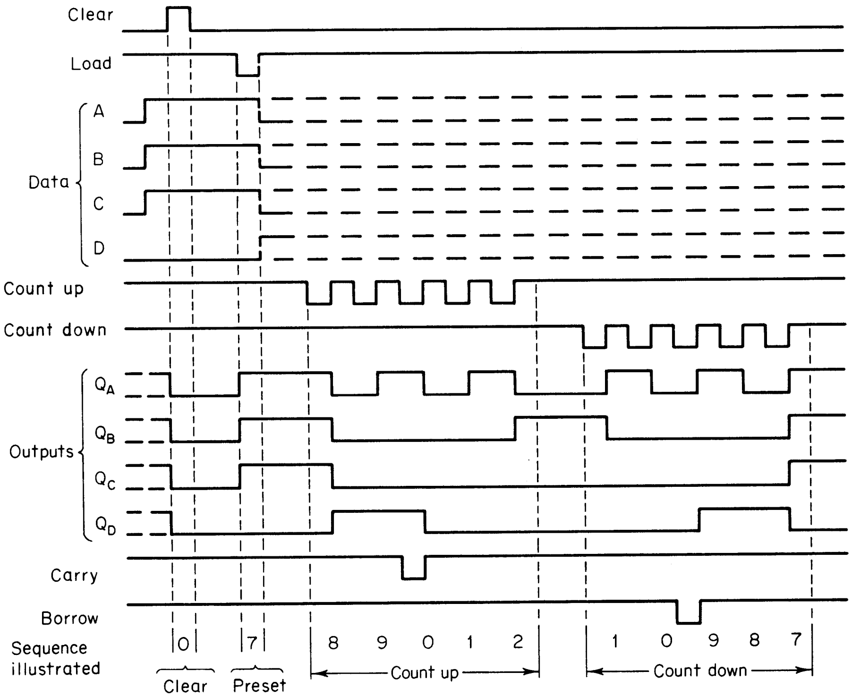

Borrow output
Carry output
Data input A
Down count
Up count
Preset
Q_A
T
$\bar{Q}_A$
Clear
Output Q_A
Data input B
Preset
Q_B
T
$\bar{Q}_B$
Clear
Output Q_B
Data input C
Preset
Q_C
T
$\bar{Q}_C$
Clear
Output Q_C
Data input D
Clear
Preset
Q_D
T
$\bar{Q}_D$
Clear
Output Q_D
Load

Figure 10.24

Illustrated below is the following sequence

1. Clear outputs to zero
2. Load (preset) to BCD seven
3. Count up to eight, nine, carry, zero, one and two
4. Count down to one, zero, borrow, nine, eight and seven

Timing diagram

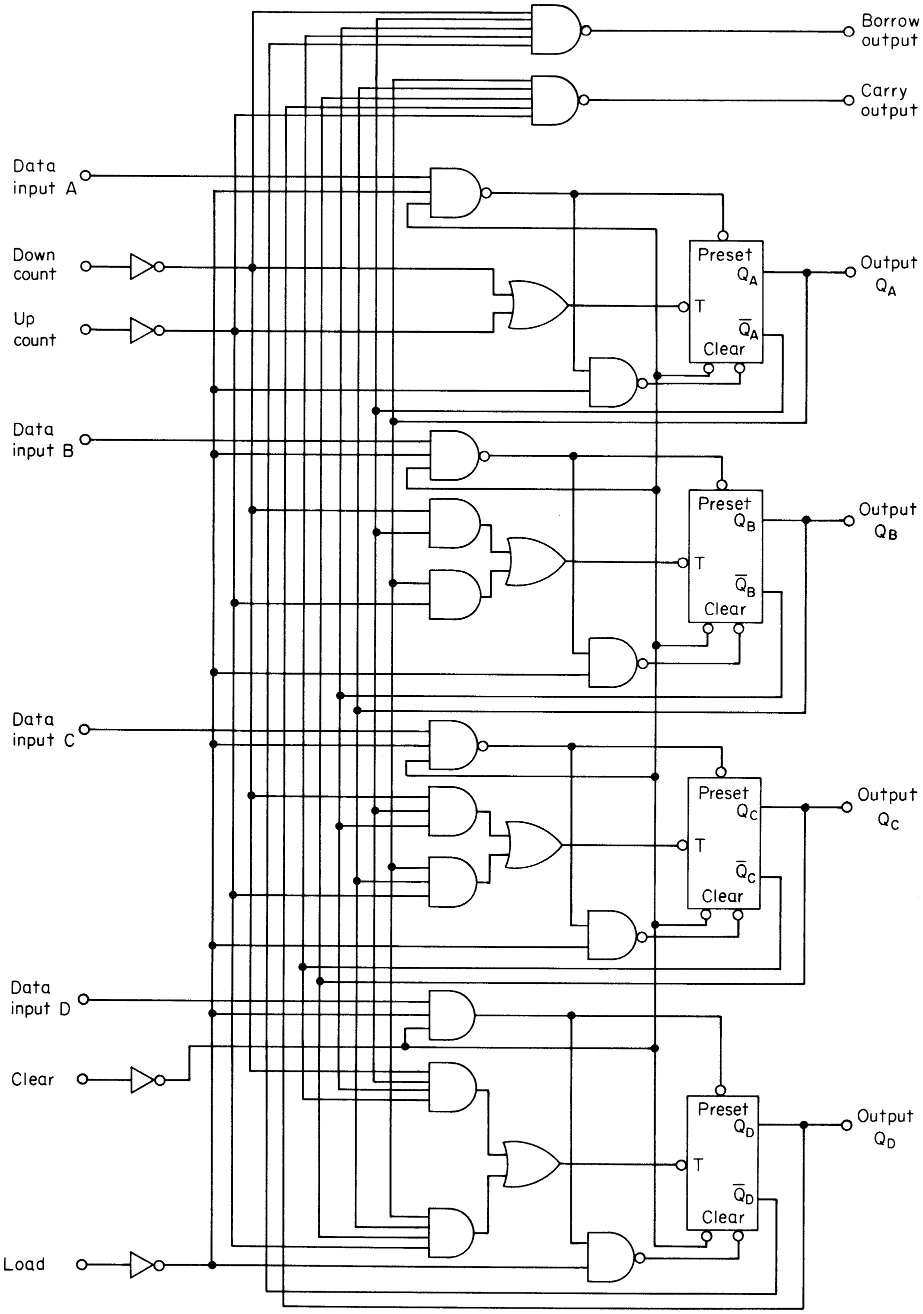

Clear
Load
Data
A
B
C
D
Count up
Count down
Outputs
Q_A
Q_B
Q_C
Q_D
Carry
Borrow
Sequence illustrated
0
7
8 9 0 1 2
1 0 9 8 7
Clear
Preset
Count up
Count down

Notes: A. Clear overrides load, data and count inputs

B. When counting up, count-down input must be high; when counting down, count-up input must be high

Figure 10.24. (*Continued*)

Logic diagram



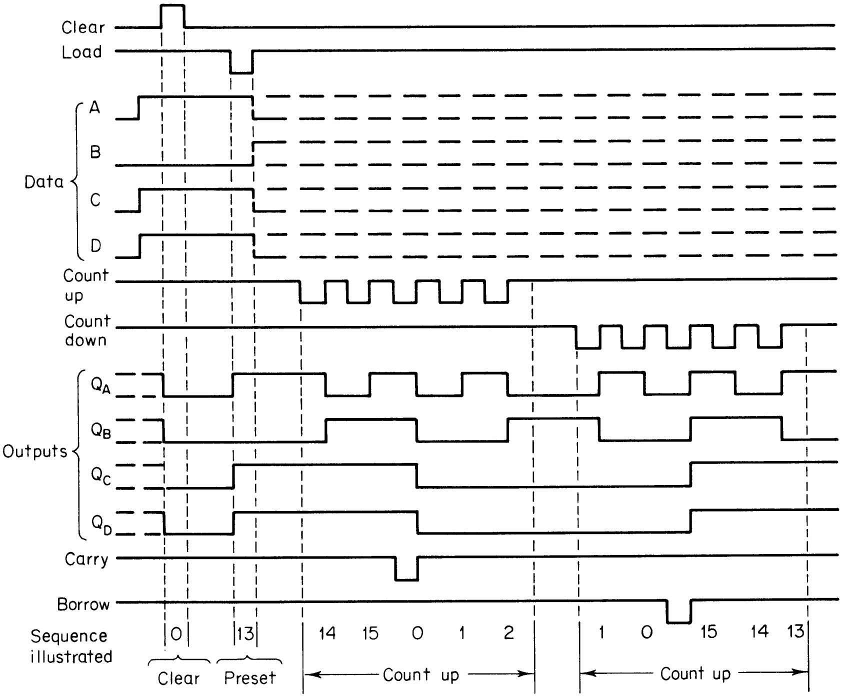
Borrow output
Carry output
Data input A
Down count
Up count
Data input B
Data input C
Data input D
Clear
Load
Preset
Q_A
T
$\overline{Q}_A$
Clear
Output Q_A
Preset
Q_B
T
$\overline{Q}_B$
Clear
Output Q_B
Preset
Q_C
T
$\overline{Q}_C$
Clear
Output Q_C
Preset
Q_D
T
$\overline{Q}_D$
Clear
Output Q_D

Fig. 10.25. SN54/74193 synchronous binary counter.

Illustrated below is the following sequence:

1. Clear output to zero
2. Load (preset) to BCD thirteen
3. Count up to fourteen, fifteen, carry, zero, one and two
4. Count down to one, zero, borrow, fifteen, fourteen, and thirteen

Timing diagram


Clear
Load
Data
A
B
C
D
Count up
Count down
Outputs
Q_A
Q_B
Q_C
Q_D
Carry
Borrow
Sequence illustrated
0
13
14 15 0 1 2
1 0 15 14 13
Clear
Preset
Count up
Count up

Notes: A. Clear overrides load, data, and count inputs

B. When counting up, count-down input must be high; when counting down, count-up input must be high

Figure 10.25. *(Continued)*

Figure 10.26 shows these counters cascaded, using ripple carry between stages. A divide-by-N circuit is shown in Fig. 10.27. One of the useful features of this circuit is that it directly interfaces with standard thumb-wheel switches for the data input.

10.4 COUNTER IMPLEMENTATION AND APPLICATIONS

Synchronous counters can be implemented by using standard line J-K flip-flops. Most control logic can be performed internally in the flip-flop since the J-K inputs of the SN7472 are 3-input NAND gates. Figures 10.28 through 10.38 show the SN54/7473 and SN54/7472 used in various binary divide-by-N counters.

Divide-by-N Counters. For some counting applications, the SN7490, SN7492, and SN7493 ripple counters may be modified to change the count cycle. By decoding any desired cycle length at the outputs of the A, B, C, and D flip-flops and

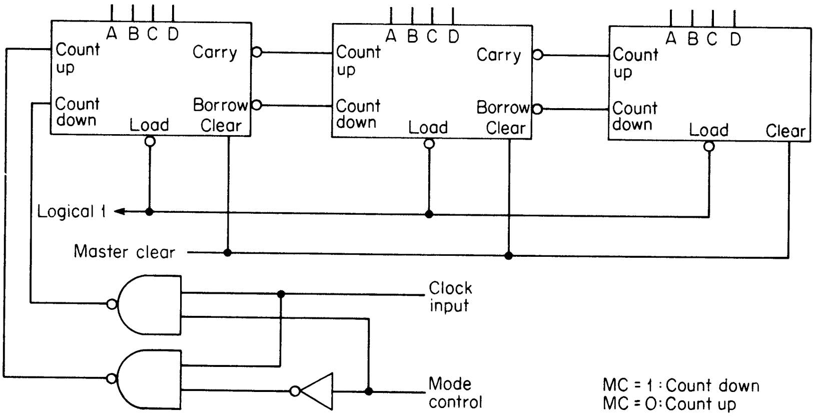
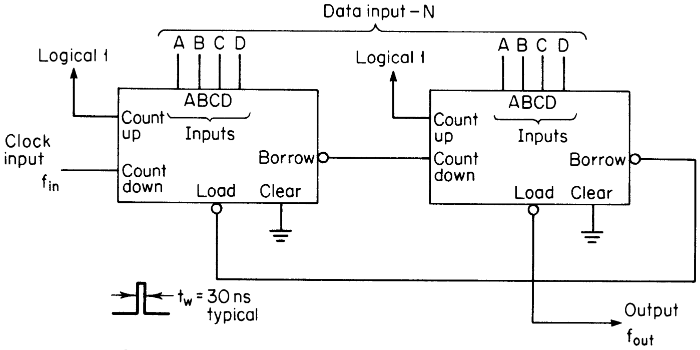

A B C D
Count up
Carry
Count down
Borrow
Load
Clear
Logical 1
Master clear
Clock input
Mode control
MC = 1 : Count down
MC = 0: Count up

Fig. 10.26. Synchronous counter with ripple carry using the SN54/74192 or SN54/74193 synchronous up/down counter.


Data input – N
A B C D
Logical 1
Clock input f_{in}
ABCD Inputs
Count up
Count down
Borrow
Load
Clear
t_w = 30 ns typical
Output f_{out}

SN54/74192 Decade counter:

$$f_o = \frac{f_{in}}{N} = \frac{f_{in}}{1A + 2B + 4C + 8D + 10E + 20F + 40G + 80H}$$

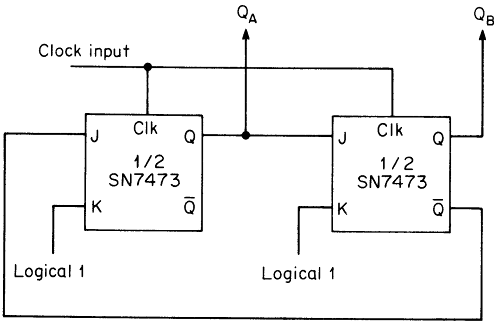
where: $1 \leq N \leq 99$; Data to each counter: 0 thru 9

SN54/74193 4-bit binary counter

$$f_o = \frac{f_{in}}{N} = \frac{f_{in}}{1A + 2B + 4C + 8D + 16E + 32F + 64G + 128H}$$

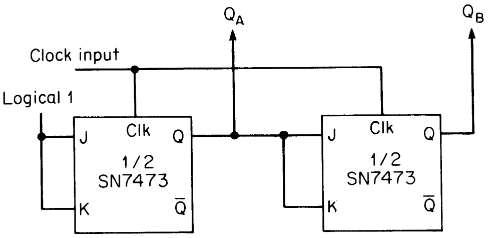
where: $1 \leq N \leq 255$

Fig. 10.27. Divide-by-*N* counter using the SN54/74192 or SN54/74193.


Q_A
Q_B
Clock input
J
Clk
Q
1/2 SN7473
K
$\bar{Q}$
Logical 1

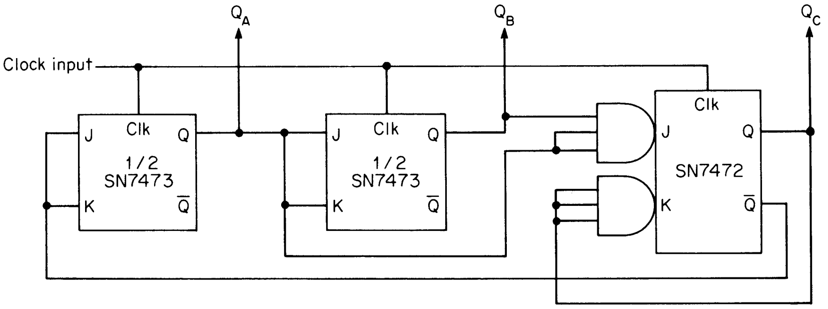
State	Q_A	Q_B
0	0	0
1	1	0
2	0	1

Fig. 10.28. Binary divide-by-3 counter.


Q_A
Q_B
Clock input
Logical 1
J
Clk
Q
1/2
SN7473
K
Q̄
J
Clk
Q
1/2
SN7473
K
Q̄

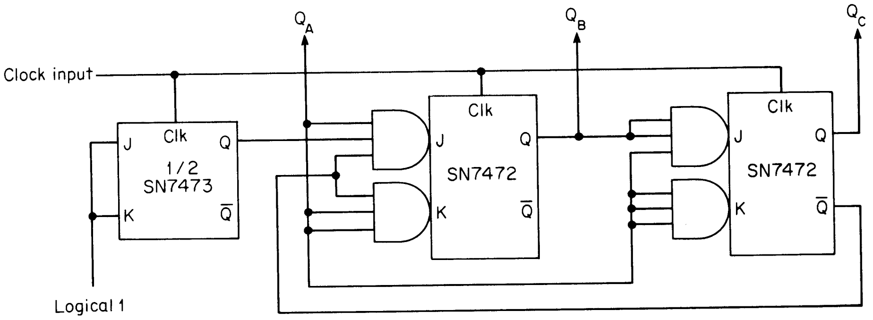
State	Q_A	Q_B
0	0	0
1	1	0
2	0	1
3	1	1

Fig. 10.29. Binary divide-by-4 counter.


Q_A
Q_B
Q_C
Clock input
J
Clk
Q
1/2
SN7473
K
Q̄
J
Clk
Q
1/2
SN7473
K
Q̄
Clk
J
Q
SN7472
K
Q̄

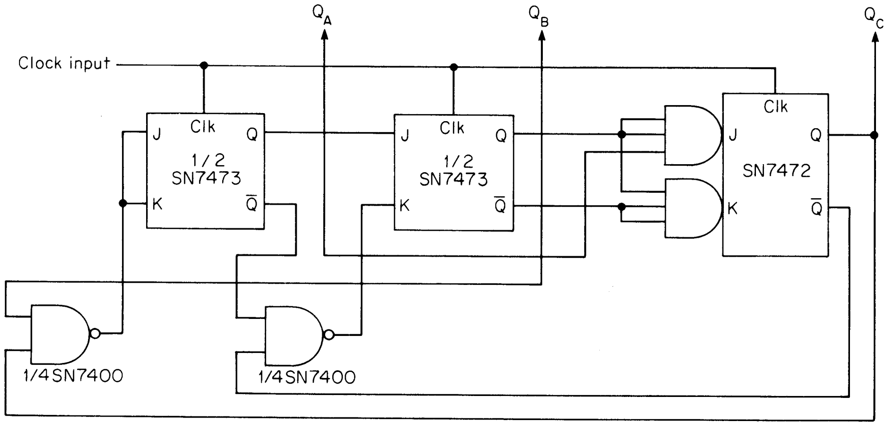
State	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1

Fig. 10.30. Binary divide-by-5 counter.


Q_A
Q_B
Q_C
Clock input
J
Clk
Q
1/2
SN7473
K
Q̄
Logical 1
Clk
J
Q
SN7472
K
Q̄
Clk
J
Q
SN7472
K
Q̄

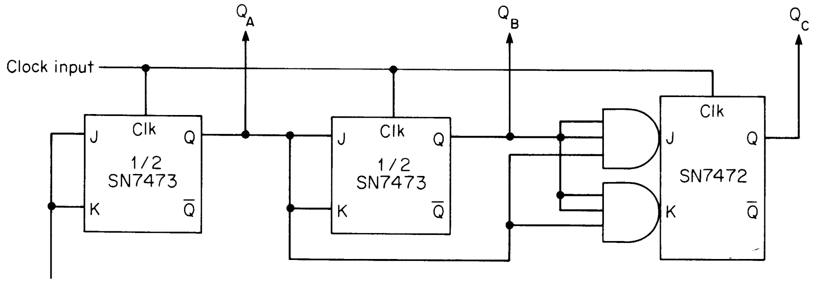
State	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1

Fig. 10.31. Binary divide-by-6 counter.


Q_A
Q_B
Q_C
Clock input
J
Clk
Q
1/2
SN7473
K
Q̄
J
Clk
Q
1/2
SN7473
K
Q̄
Clk
J
Q
SN7472
K
Q̄
1/4SN7400
1/4SN7400

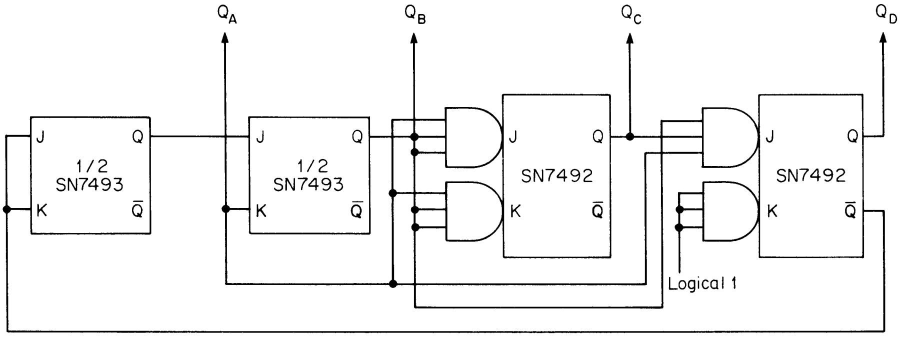
State	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1

Fig. 10.32. Binary divide-by-7 counter.


Q_A
Q_B
Q_C
Clock input
J
Clk
Q
1/2
SN7473
K
Q̄
J
Clk
Q
1/2
SN7473
K
Q̄
Clk
J
Q
SN7472
K
Q̄

State	Q_A	Q_B	Q_C
0	0	0	0
1	1	0	0
2	0	1	0
3	1	1	0
4	0	0	1
5	1	0	1
6	0	1	1
7	1	1	1

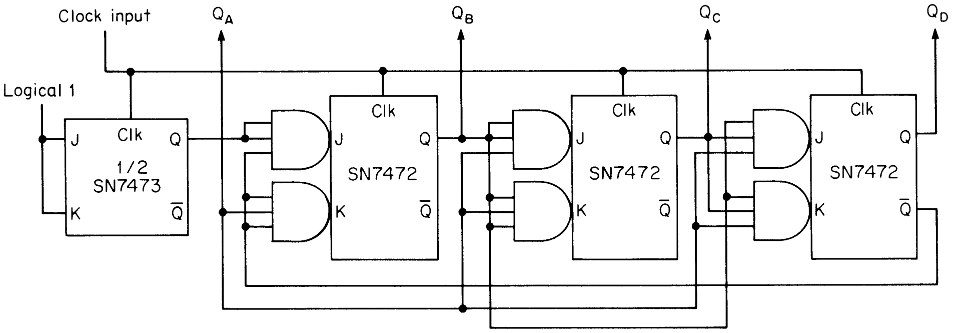
Fig. 10.33. Binary divide-by-8 counter.



Q_A
Q_B
Q_C
Q_D
J
Q
1/2
SN7493
K
$\bar{Q}$
J
Q
1/2
SN7493
K
$\bar{Q}$
J
Q
SN7492
K
$\bar{Q}$
J
Q
SN7492
K
$\bar{Q}$
Logical 1

State	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1

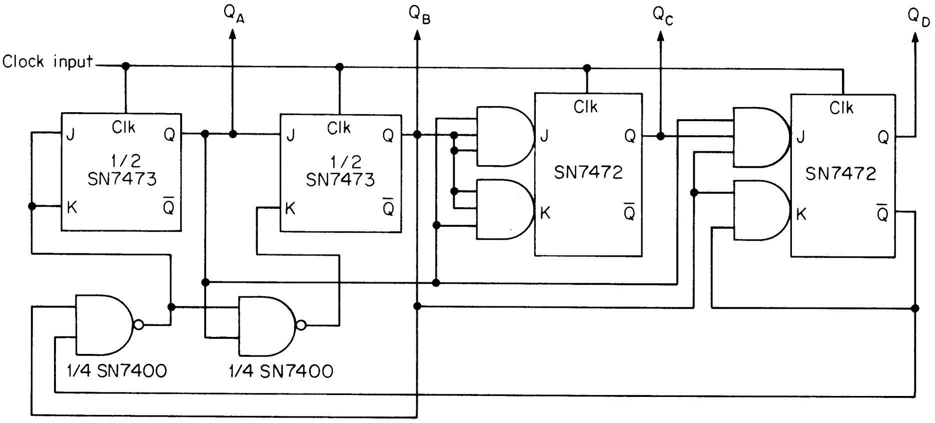
Fig. 10.34. Binary divide-by-9 counter.



Clock input
Q_A
Q_B
Q_C
Q_D
Logical 1
J
Clk
Q
1/2
SN7473
K
$\bar{Q}$
Clk
J
Q
SN7472
K
$\bar{Q}$
Clk
J
Q
SN7472
K
$\bar{Q}$
Clk
J
Q
SN7472
K
$\bar{Q}$

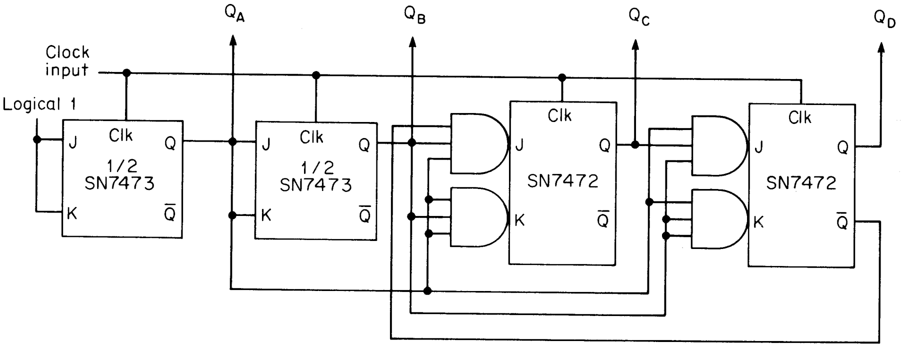
State	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1

Fig. 10.35. Binary divide-by-10 counter.


Q_A
Q_B
Q_C
Q_D
Clock input
J
Clk
Q
1/2
SN7473
K
$\overline{Q}$
J
Clk
Q
1/2
SN7473
K
$\overline{Q}$
Clk
J
Q
SN7472
K
$\overline{Q}$
Clk
J
Q
SN7472
K
$\overline{Q}$
1/4 SN7400
1/4 SN7400

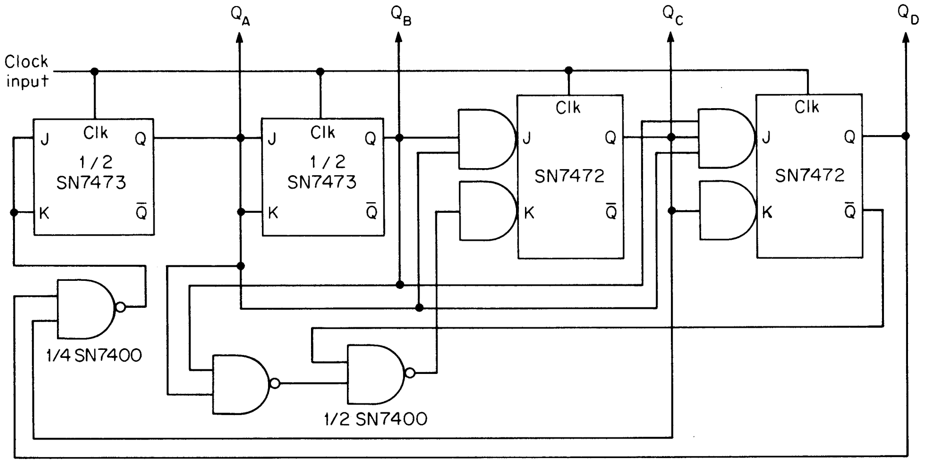
State	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1

Fig. 10.36. Binary divide-by-11 counter.


Q_A
Q_B
Q_C
Q_D
Clock input
Logical 1
J
Clk
Q
1/2
SN7473
K
$\overline{Q}$
J
Clk
Q
1/2
SN7473
K
$\overline{Q}$
Clk
J
Q
SN7472
K
$\overline{Q}$
Clk
J
Q
SN7472
K
$\overline{Q}$

State	Q	Q	Q	Q
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1

Fig. 10.37. Binary divide-by-12 counter.



State	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0
6	0	1	1	0
7	1	1	1	0
8	0	0	0	1
9	1	0	0	1
10	0	1	0	1
11	1	1	0	1
12	0	0	1	1

Fig. 10.38. Binary divide-by-13 counter.

feeding this signal to the asynchronous clear inputs, the counter can be made to operate with a shortened cycle. The maximum frequency in this mode is limited by the restriction that the clock should not go to a logical 0 less than 40 ns after the counter has been reset to 0 (see Fig. 10.39). The outputs that are in a logical 1 state at the count *N* are fed into the $R_{0(1)}$ and $R_{0(2)}$ inputs. The counter will then reset to the all-0 state when the count *N* appears at these outputs.

According to the count cycle *N*, voltage spikes may appear on some output lines. See output *B* waveform (Fig. 10.40). When the outputs are to be decoded, a strobe gate should be provided to inhibit false output data (Fig. 10.41). When the outputs are not equally and heavily loaded, timing difficulties may be encountered at temperature extremes. Buffering outputs with the configuration shown in Fig. 10.46 will prevent these difficulties. Figures 10.42 through 10.52 show how the SN54/7492 and SN54/7493 may be used to design various binary divide-by-*N* counters.

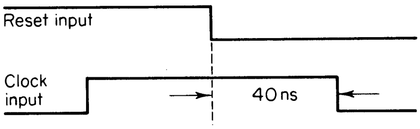
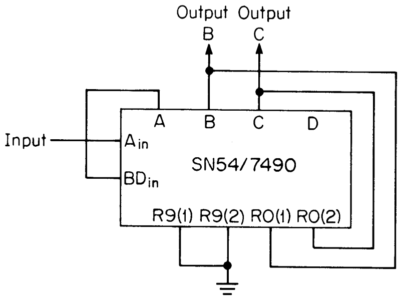


Fig. 10.39. Typical reset timing requirement.


Output B
Output C
Input
A
B
C
D
Ain
BDin
SN54/7490
R9(1) R9(2) RO(1) RO(2)

Clock pulse	Output			
	A	B	C	D
t_n	0	0	0	0
t_{n+1}	1	0	0	0
t_{n+2}	0	1	0	0
t_{n+3}	1	1	0	0
t_{n+4}	0	0	1	0
t_{n+5}	1	0	1	0
t_{n+6}	0/0	1/0	1/0	0/0

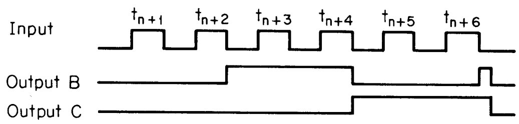

Input
tn+1
tn+2
tn+3
tn+4
tn+5
tn+6
Output B
Output C

Fig. 10.40. Binary divide-by-6 ripple counter using SN7490.

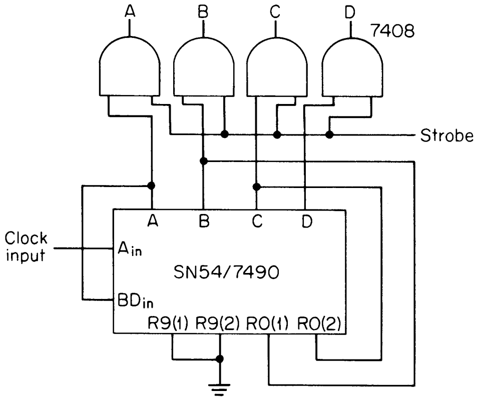

A
B
C
D
7408
Strobe
Clock input
A
B
C
D
Ain
BDin
SN54/7490
R9(1) R9(2) RO(1) RO(2)

Fig. 10.41. Binary divide-by-6 ripple counter using SN7490 (buffered outputs).

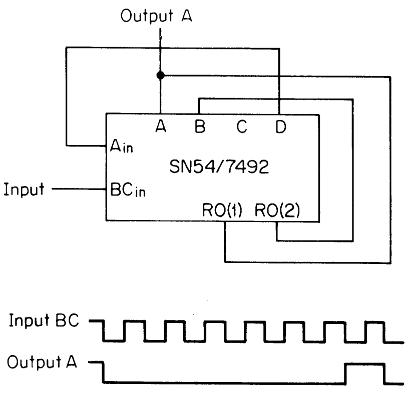

Output A
A
B
C
D
Ain
SN54/7492
Input
BCin
RO(1) RO(2)
Input BC
Output A

Fig. 10.42. Divide-by-7 ripple counter using SN7492.

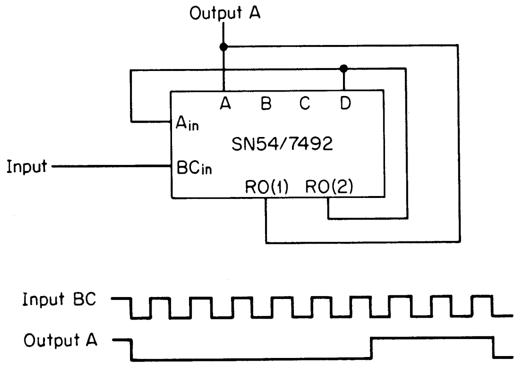

Output A
A B C D
Ain
SN54/7492
Input
BCin
RO(1) RO(2)
Input BC
Output A

Fig. 10.43. Divide-by-9 ripple counter using SN7492.

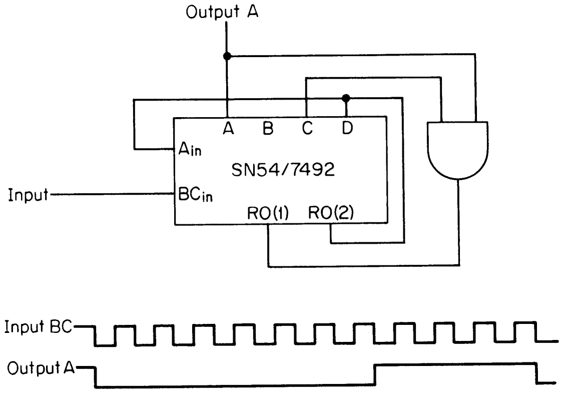

Output A
A B C D
Ain
SN54/7492
Input
BCin
RO(1) RO(2)
Input BC
Output A

Fig. 10.44. Divide-by-11 ripple counter using SN7492.

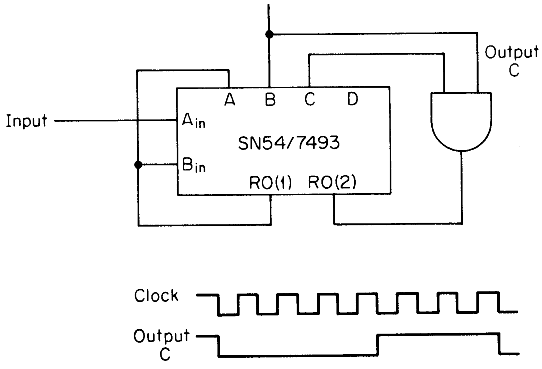

Output C
A B C D
Input
Ain
SN54/7493
Bin
RO(1) RO(2)
Clock
Output C

Fig. 10.45. Divide-by-7 ripple counter using SN54/7493.

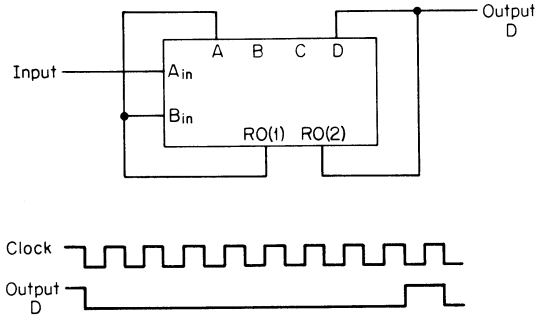

Output D
A B C D
Input
Ain
Bin
RO(1) RO(2)
Clock
Output D

Fig. 10.46. Binary divide-by-9 ripple counter using SN54/7493.

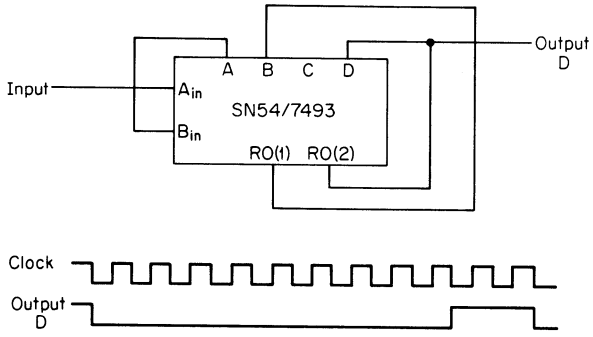

Input
A
B
C
D
Ain
SN54/7493
Bin
RO(1)
RO(2)
Output
D
Clock
Output
D

Fig. 10.47. Binary divide-by-10 ripple counter using SN54/7493.

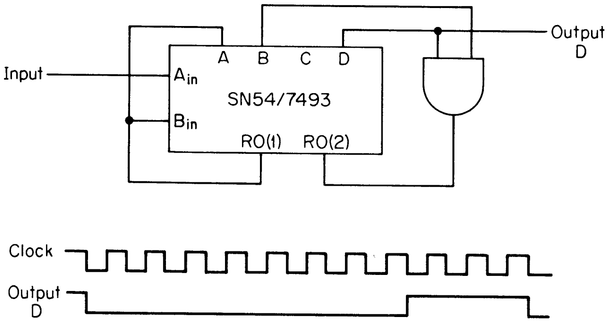

Input
A
B
C
D
Ain
SN54/7493
Bin
RO(1)
RO(2)
Output
D
Clock
Output
D

Fig. 10.48. Binary divide-by-11 ripple counter using SN54/7493.

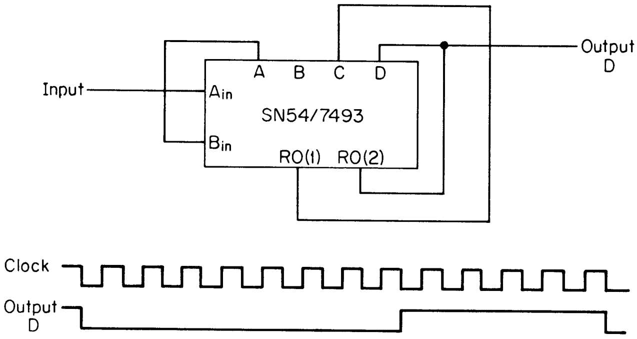
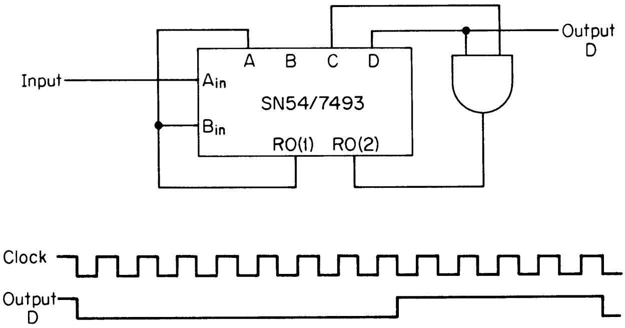
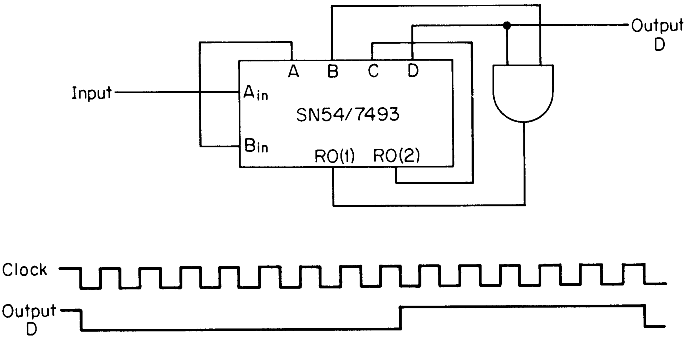

Input
A
B
C
D
Ain
SN54/7493
Bin
RO(1)
RO(2)
Output
D
Clock
Output
D

Fig. 10.49. Binary divide-by-12 ripple counter using SN54/7493.



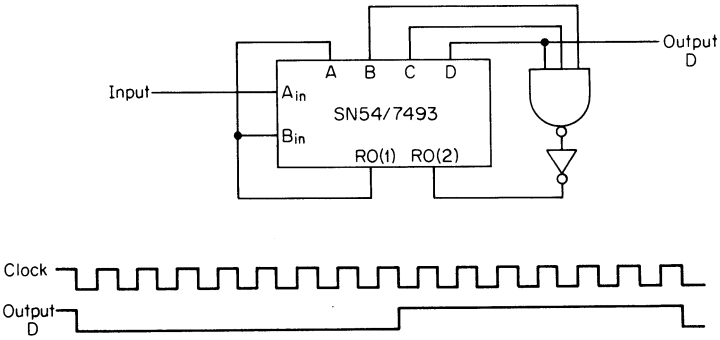
Output
D
A B C D
Input
Ain
SN54/7493
Bin
RO(1) RO(2)
Clock
Output
D

Fig. 10.50. Binary divide-by-13 ripple counter using SN54/7493.



Output
D
A B C D
Input
Ain
SN54/7493
Bin
RO(1) RO(2)
Clock
Output
D

Fig. 10.51. Binary divide-by-14 ripple counter using SN54/7493.



Output
D
A B C D
Input
Ain
SN54/7493
Bin
RO(1) RO(2)
Clock
Output
D

Fig. 10.52. Binary divide-by-15 ripple counter using SN54/7493.

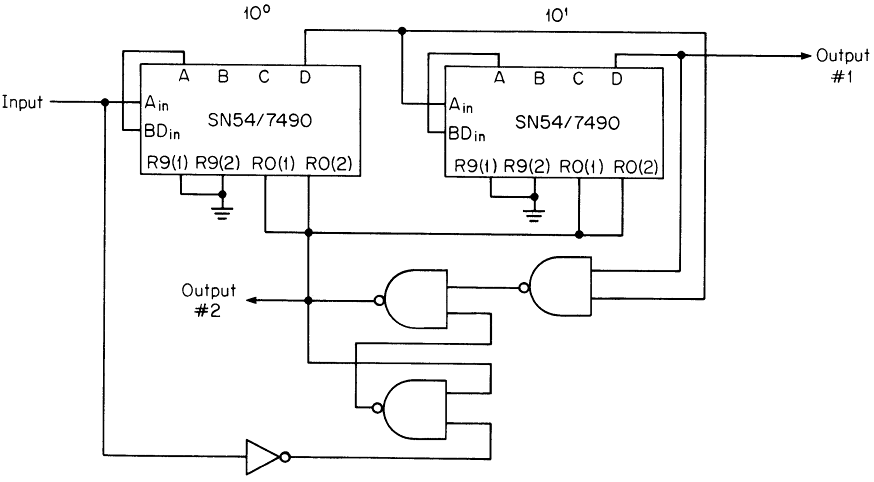
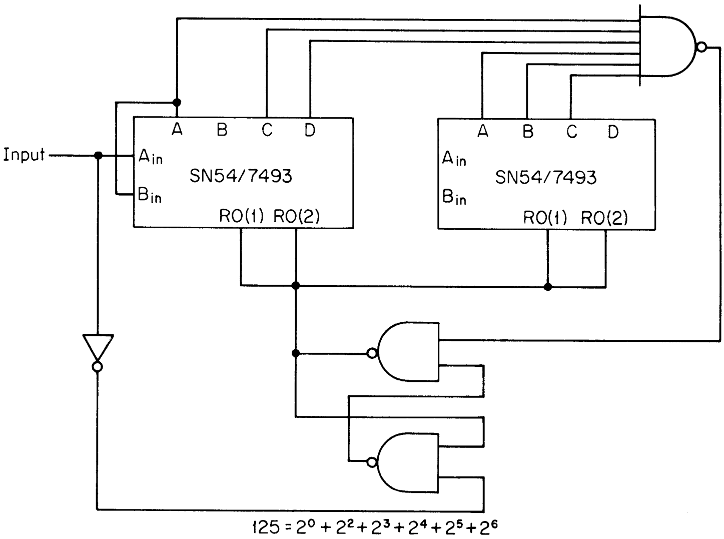

10^0
10^1
Input
A B C D
A_{in}
BD_{in}
SN54/7490
R9(1) R9(2) RO(1) RO(2)
Output #1
Output #2

Fig. 10.53. BCD divide-by-88 counter using SN54/7490.

By cascading SN54/7490, SN54/7492, or SN54/7493 devices together, ripple counters of any cycle length can be created.

For division by a large *N* when a binary count is desired, the counter packages may be cascaded like a ripple counter, as in Fig. 10.53. For each complete cycle of counter *A*, *B* is incremented by 1. The desired cycle length is then decoded and used to reset both counters to 0. Depending upon the cycle length, false outputs may be generated by the NAND gate. It should be strobed when necessary to eliminate false data. Another example using SN54/7493s is shown in Fig. 10.54.


Input
A B C D
A_{in}
B_{in}
SN54/7493
RO(1) RO(2)

$125 = 2^0 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6$

Fig. 10.54. Binary divide-by-125 counter using SN54/7493.

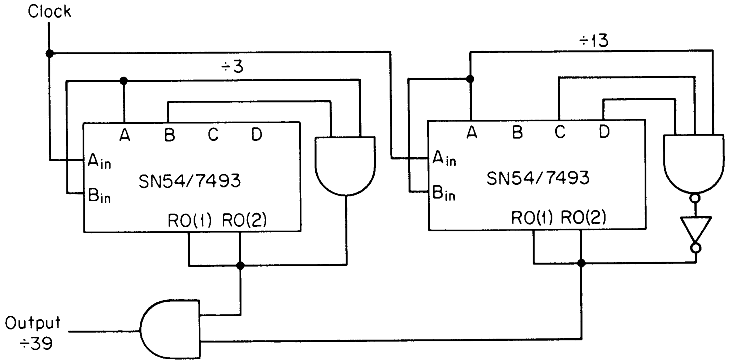


Fig. 10.55. Divide-by-39 counter with SN54/7493s using relative prime number technique.

Another method of obtaining large frequency-division ratios, as long as the counting sequence is not significant, is to count in a relative prime number system. One counter is set up to divide by some N (2, 3, 4, . . . , 15). The second counter divides by some integer that is a relative prime of the first. Counters are combined in parallel as shown in Fig. 10.55, and the resultant division is the product of division done by each counter. This can also be accomplished by a series cascade connection between the divide-by-3 system and the divide-by-13 system. The advantage of the relative prime number technique is less phase lag between the output pulse and the clock pulse.

11

Shift Registers

A shift register is a group of cascaded flip-flops. Each flip-flop output is connected to the input of the following flip-flop, and a common clock pulse is applied to all flip-flops, clocking them synchronously.

In the basic shift register, the only external control input other than from the clock is to the input of the first flip-flop. There is always an output from the last stage, and access may or may not be provided to the outputs of the other stages. The input to a shift register may be single-rail or double-rail, and the double-rail input may be either a *J-K* or *R-S* input. Figure 11.1 is a generalized shift register with double- and single-rail inputs. Modifications of the basic shift register shown in Fig. 11.1 include capability of parallel loading, exemplified by the SN54/7495,

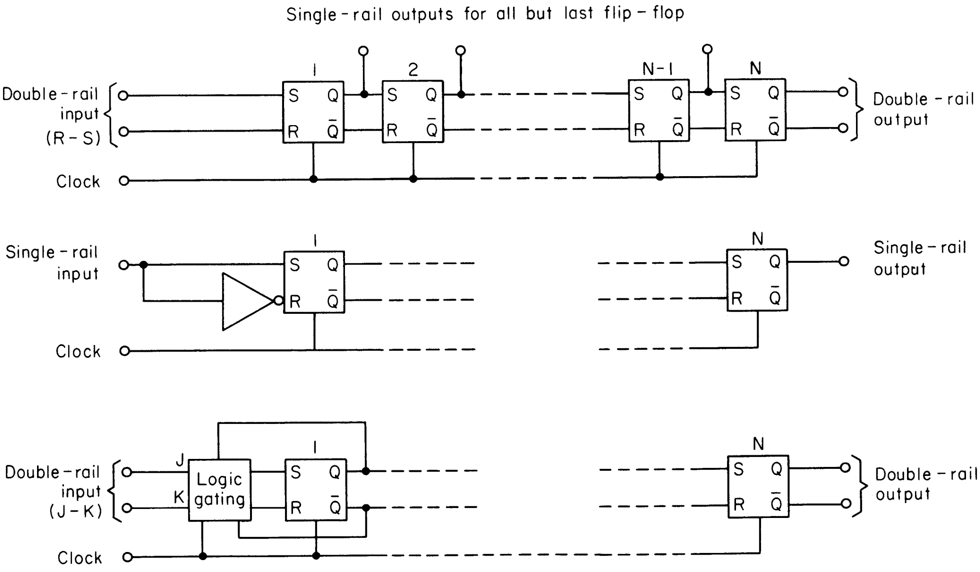

Single-rail outputs for all but last flip-flop
Double-rail input (R-S)
Clock
1
2
N-1
N
S
Q
R
Q̄
Double-rail output
Single-rail input
Single-rail output
Double-rail input (J-K)
J
K
Logic gating
Double-rail output

Fig. 11.1. Generalized basic shift register, showing various input/output types.

Table 11.1. Comparison of TTL Shift Registers

Type no.	Bit length	First-stage input*	Last-stage output*	Parallel load	All FF outputs available	Clear	Preset	Power dissipation†	Clock rate, MHz‡
Standard series									
SN54/7491A	8	SR	DR	No	No	No	No	175	18
SN54/7494	4	SR	SR	No	No	Yes	Yes	175	15
SN54/7495	4	SR	SR	Yes	Yes	No	No	250	31
SN54/7496	5	SR	SR	No	Yes	Yes	Yes	240	15
Low-power series									
SN54/74L91	8	SR	DR	No	No	No	No	17.5	6.5
SN54/74L95	4	SR	SR	Yes	Yes	No	No	19	5
SN54/74L98	4	SR	SR	Yes	Yes	No	No	25	5
SN54/74L99	4	DR	DR	Yes	Yes	No	No	19	5

*SR = single rail; DR = double rail.
†Typical average power dissipation at 5.0 V.
‡Typical maximum clock rate.

SN54/74L95, and SN54/74L99, and provision for preset and clear operations, found in the SN54/7494 and SN54/7496. These items will be discussed in more detail as they apply to a particular integrated-circuit package. Table 11.1 is provided to allow easy comparison of some of the important features of Series 54/74 TTL shift registers, both standard and low-power.

11.1 SN54/74 SHIFT REGISTERS

This section presents brief descriptions of each of the Series 54/74 shift register types, pointing out some of the less obvious implications of performance specifications.

8-bit Shift Registers. The SN54/7491A and SN54/74L91 are 8-bit shift registers of the basic type described earlier. The single-rail input is gated and is controlled by inputs *A* and *B*. Figure 11.2 is a functional block diagram of this IC. Data are transferred on the positive-going edge of the clock pulse.

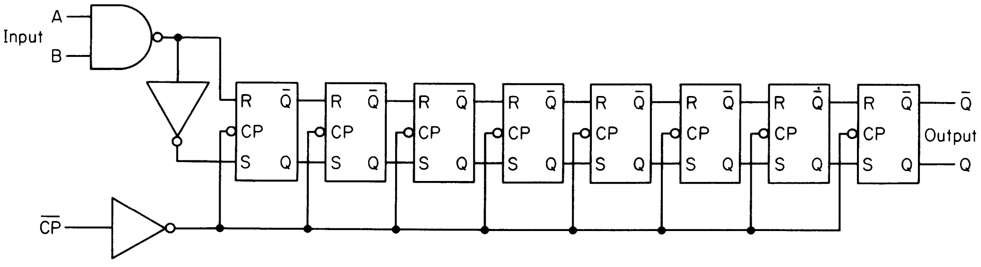


Fig. 11.2. SN54/7491A and SN54/74L91: functional block diagram.

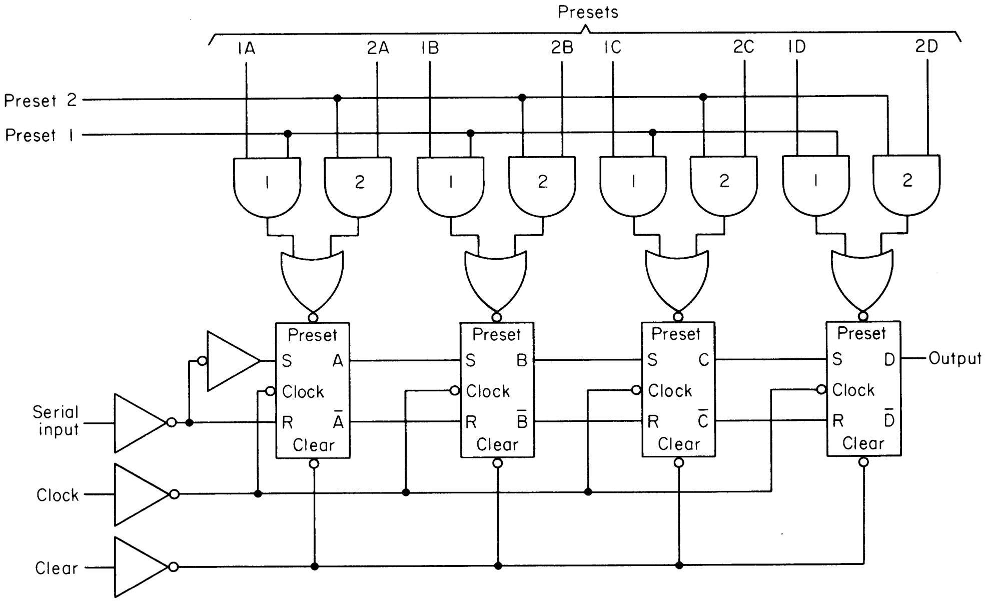
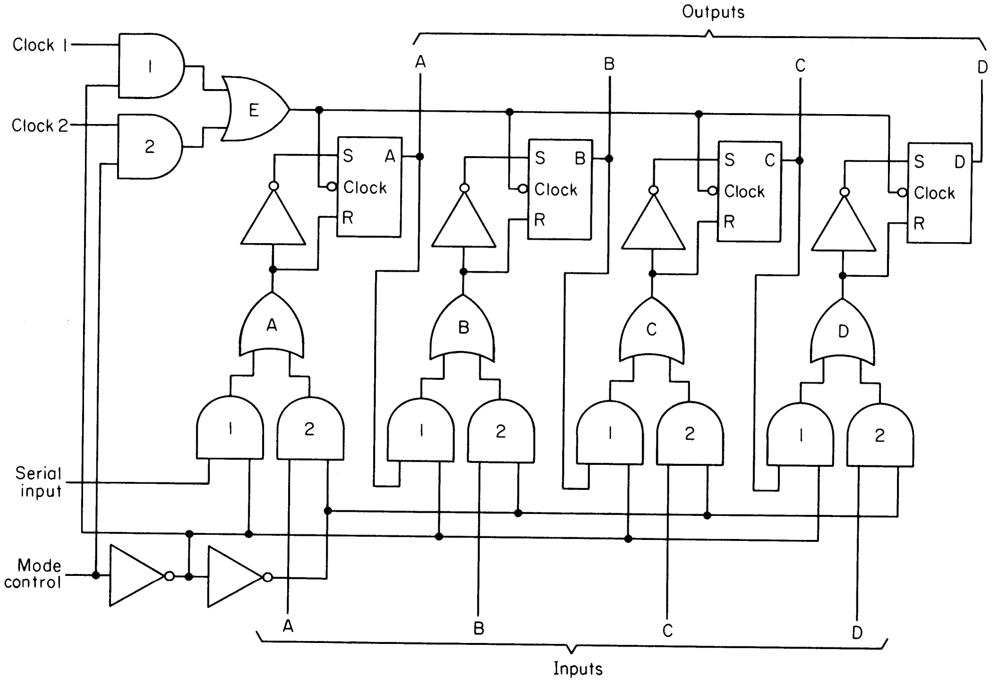


Fig. 11.3. SN54/7494: functional block diagram.

4-bit Registers; Parallel-in, Series-out. The SN54/7494 are basic shift registers, except that each flip-flop has provision for preset and clear operations. The clear is applied nonselectively to all flip-flops, but the preset employs additional gating so that either of two sources may be selected to preset each flip-flop individually. Notice that to perform a parallel-load operation using the preset, the preset operation must be preceded by a clear operation (setting all flip-flops to 0), since the preset can only set a flip-flop to a 1. Figure 11.3 is a functional block diagram of this device. Data are transferred on the positive-going edge of the clock pulse.

4-bit Right-shift Left-shift Registers. SN54/7495 and SN54/74L95 are 4-bit shift registers with parallel-load capability, and with all flip-flop outputs available. This makes it possible to perform the right-shift or left-shift operation under control of the mode control input. For greater flexibility, the mode control selects clock 1 for the right-shift mode and clock 2 for the parallel-load (left-shift) mode. (See Fig. 11.4.) The clock 1 and clock 2 inputs are tied together if only one clock source is required. Data transfer occurs on the negative-going edge of the clock pulse. Figure 11.5 is a functional block diagram of this device with connections indicated for selectable right-and-left-shift operation.

The interaction between the mode control and the clock inputs must be considered when using this device (this also applies to the SN54/74L99). Figure 11.6 is a functional logic drawing of the clock selection logic which is under control of the mode control input. Notice that if the nonselected clock is at 0, the selected clock is at a 1, and the mode control changes state; notice also that the clock line will change to a 0, causing the flip-flops to trigger. In many cases, it is undesirable to have the shift register clocked when the mode control changes state. The times


Outputs
A
B
C
D
Clock 1
Clock 2
1
2
E
S A
Clock
R
S B
Clock
R
S C
Clock
R
S D
Clock
R
A
B
C
D
1
2
1
2
1
2
1
2
Serial input
Mode control
A
B
C
D
Inputs

MC = 0 for right shift
MC = 1 for parallel load or left shift

Fig. 11.4. SN54/7495 and SN54/74L95: functional block diagram.

t_3 and t_4 specify the amount of delay before the previously selected clock can become a 1 (with respect to the mode control) following a state change of the mode control. For all these circuits, the time t_3 is 0. The logic diagram of Fig. 11.6 shows why: the mode control and clock 2 go directly to AND gate 2, so there is no delay between them within the IC package. Notice that there is propagation delay, caused by the inversion of the mode control signal into AND gate 1. The time t_4 is specified at 10 or 100 ns, depending on the IC, to allow for the propagation of the inverter. These restrictions do allow the mode control and clock 2 inputs to be tied together.

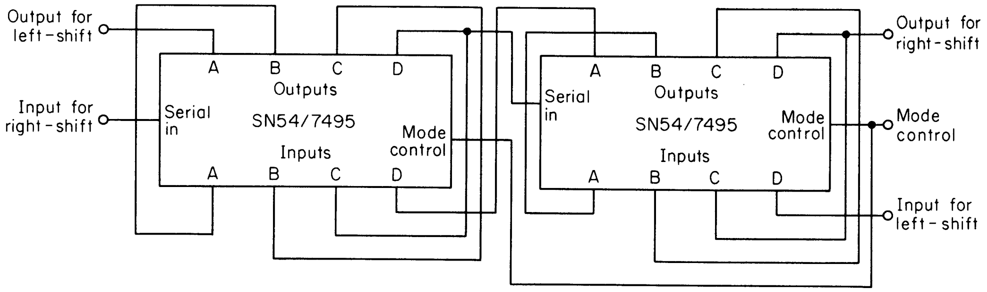
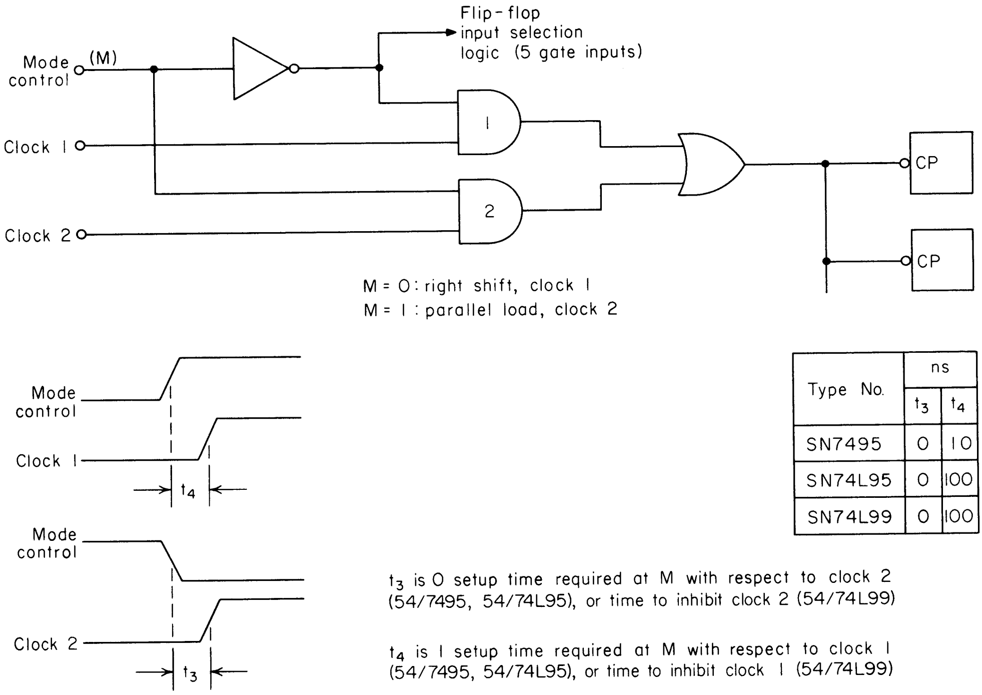

Output for left-shift
Input for right-shift
A
B
C
D
Outputs
Serial in
SN54/7495
Mode control
Inputs
A
B
C
D
A
B
C
D
Outputs
Serial in
SN54/7495
Mode control
Inputs
A
B
C
D
Output for right-shift
Mode control
Input for left-shift

Fig. 11.5. 8-bit right or left shift register using SN54/7495, SN54/74L95.


Flip-flop input selection logic (5 gate inputs)
Mode control (M)
Clock 1
Clock 2
1
2
CP
CP
M = 0: right shift, clock 1
M = 1: parallel load, clock 2
Mode control
Clock 1
t_4
Mode control
Clock 2
t_3

Type No.	ns	
	t_3	t_4
SN7495	0	10
SN74L95	0	100
SN74L99	0	100

t_3 is 0 setup time required at M with respect to clock 2 (54/7495, 54/74L95), or time to inhibit clock 2 (54/74L99)

t_4 is 1 setup time required at M with respect to clock 1 (54/7495, 54/74L95), or time to inhibit clock 1 (54/74L99)

Fig. 11.6. Timing relations: mode control and clocks for SN54/7495, SN54/74L95, and SN54/74L99.

This is useful where data are parallel-loaded into the shift register. A 1-level clock pulse with this connection selects parallel load, and when the pulse goes to 0, the parallel data are clocked into the shift register.†

The SN54/74L99 are similar to the SN54/74L95 in that they have parallel-load capability and may be connected to perform right/left shift operations, but they have some additional useful features. The *J*-$\overline{K}$ input to the first flip-flop permits the first stage to act as a *J*-$\overline{K}$, *D*, or *T* flip-flop, and avoids additional gating for the implementation of a Johnson counter. In addition, the $\overline{Q}_D$ (complement of the last flip-flop is also an output. Refer to the discussion of the SN54/7495 and SN54/74L95 and Fig. 11.6 for notes on the interaction of the mode control and clock inputs. Figure 11.9 is a functional block diagram for the SN54/74L99.

5-bit Shift Registers. The SN54/7496, like the SN54/7494, have preset and clear capability. The clear is common to all flip-flops, but the preset can be selectively applied to each flip-flop. To perform a parallel-load operation, a clear operation must be performed before a preset operation, as is the case with SN54/7494. Data transfer occurs on the positive-going edge of the clock pulse. Figure 11.7 shows a functional block diagram of SN54/7496.

---

†See also S. W. Golomb, "Shift Register Sequences," Holden-Day, Inc., Publisher, San Francisco, 1967.

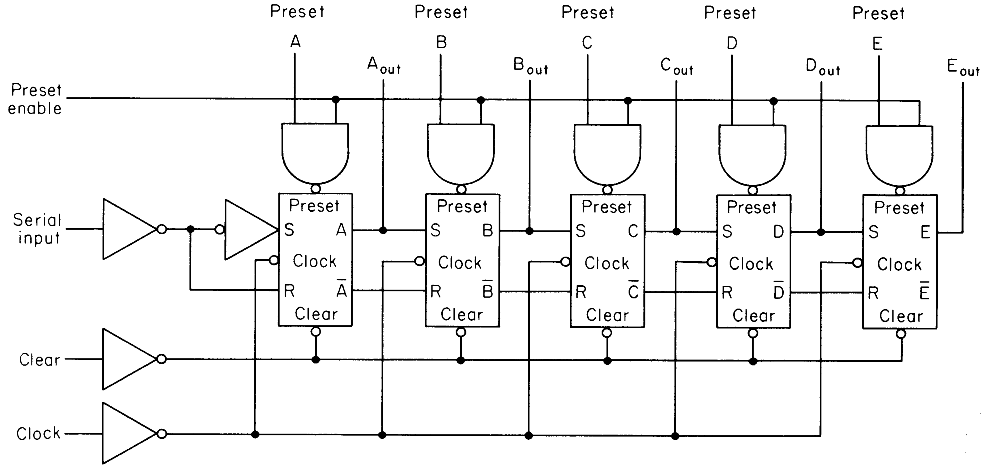

Preset A
Preset B
Preset C
Preset D
Preset E
A_{out}
B_{out}
C_{out}
D_{out}
E_{out}
Preset enable
Serial input
Clear
Clock
Preset
S
R
Clock
Clear
A
$\bar{A}$
B
$\bar{B}$
C
$\bar{C}$
D
$\bar{D}$
E
$\bar{E}$

Fig. 11.7. SN54/7496: functional block diagram.

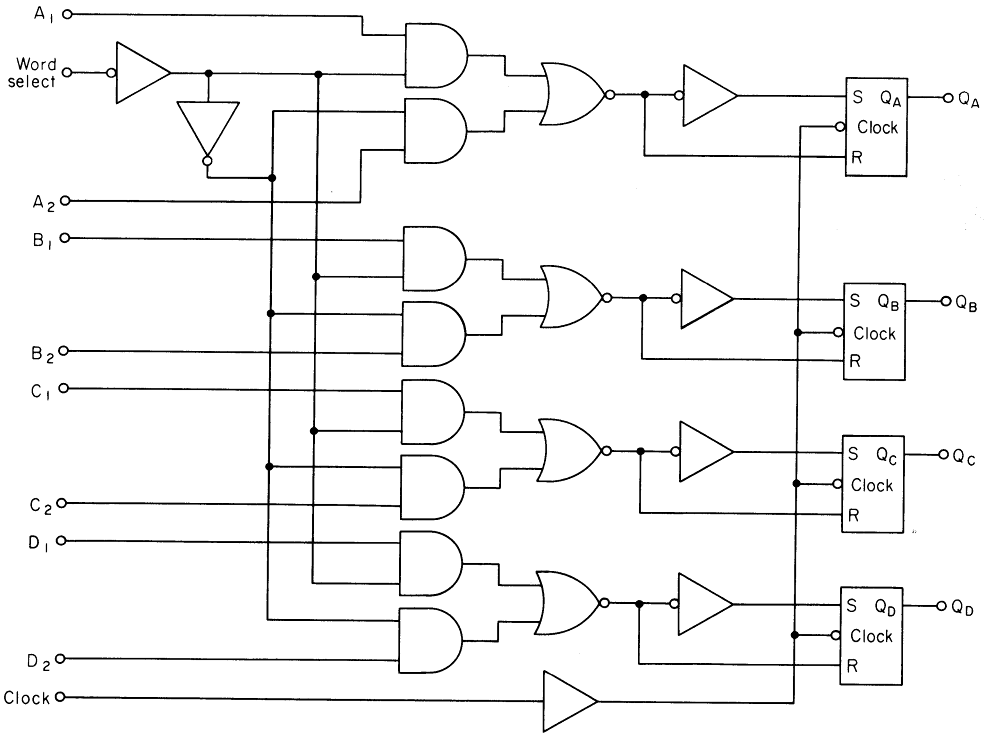

A_1
Word select
A_2
B_1
B_2
C_1
C_2
D_1
D_2
Clock
S
Clock
R
Q_A
Q_B
Q_C
Q_D

Figure 11.8

4-bit Data Selectors/Storage Registers. Although SN54/74L98 are not actually shift registers, they are included in this section because they are quite similar to shift registers and can readily be used as shift registers if the proper circuit connections are made. Figure 11.8 is the functional block diagram. The selected word data are transferred to the flip-flop outputs on the negative-going edge of the clock pulse. When the word select input is a 0, word 1 is applied to the flip-flop inputs, and when the word select input is at a 1, word 2 is selected.

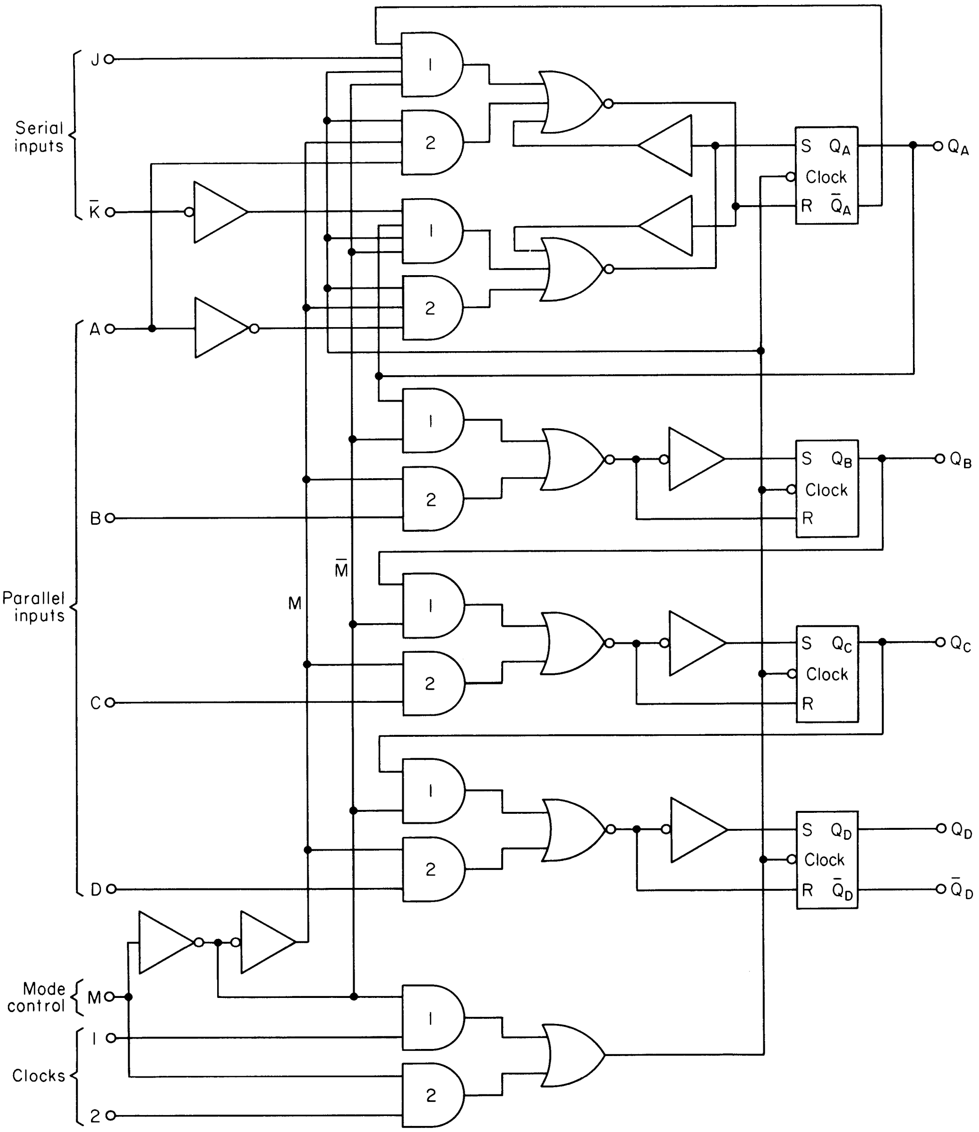

Serial inputs
J
K̄
A
B
C
D
Parallel inputs
Mode control
M
Clocks
1
2
M
M̄
1
2
S
Q_A
Clock
R
Q̄_A
Q_A
Q_B
Q_C
Q_D
Q̄_D
S
Q_B
Clock
R
S
Q_C
Clock
R
S
Q_D
Clock
R
Q̄_D

Fig. 11.9. SN54/74L99: functional block diagram.

11.2 SHIFT REGISTER COUNTERS AND GENERATORS

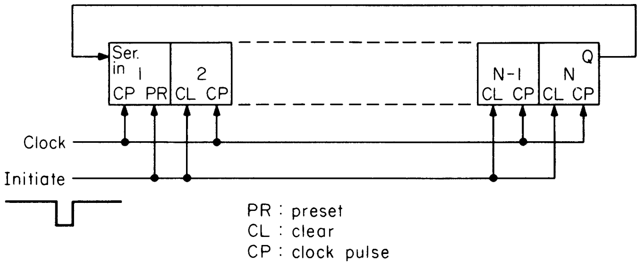
The common denominator for this class of circuits is that feedback from the flip-flop outputs of the shift register are fed back to control the input of the shift register (first-stage flip-flop).

Ring Counters. A basic ring counter is shown in Fig. 11.10. A ring counter has *N* states where *N* is the bit length or number of stages of the shift register. A ring counter operates by circulating either a single 1 or 0 level. This feature means that the counter states are essentially decoded without any additional logic. The fact that additional logic is not needed is the principal reason for using a ring counter, which is otherwise an inefficient device because it does not make efficient use of the total states available.

The ring counter of Fig. 11.10 circulates a 1, and it must be started by the initiate pulse when power is applied. The initiate pulse presets a 1 into the first stage and clears all others. In the event of power failure or noise, this counter must have an initiate pulse applied to avoid invalid states.

Figure 11.11 is an example of a 4-bit ring counter. Figure 11.11*a* is the simple feedback case, with a state diagram for the four valid states and the remaining twelve invalid states. Figure 11.11*b* presets a more complex feedback arrangement that ensures return to the valid states within a maximum of four clock pulses. The feedback logic applies a 1 to the first stage only when the *A*, *B*, and *C* stages are 0, so that preset and clear operations are not necessary if the time is available for the counter to self-correct. The state diagram is modified as shown in Fig. 11.11*b*, indicating all possible sequences for the counter.

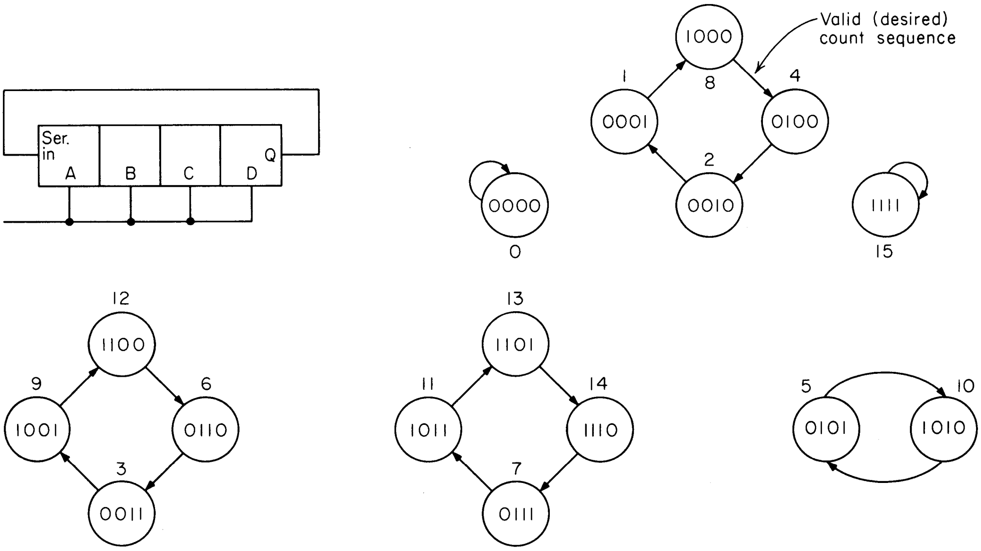
Figure 11.12 shows a 5-bit ring counter using an SN54/7496 shift register that circulates a 0. The clear and preset are shown deactivated, but they can readily be used to initiate if necessary.



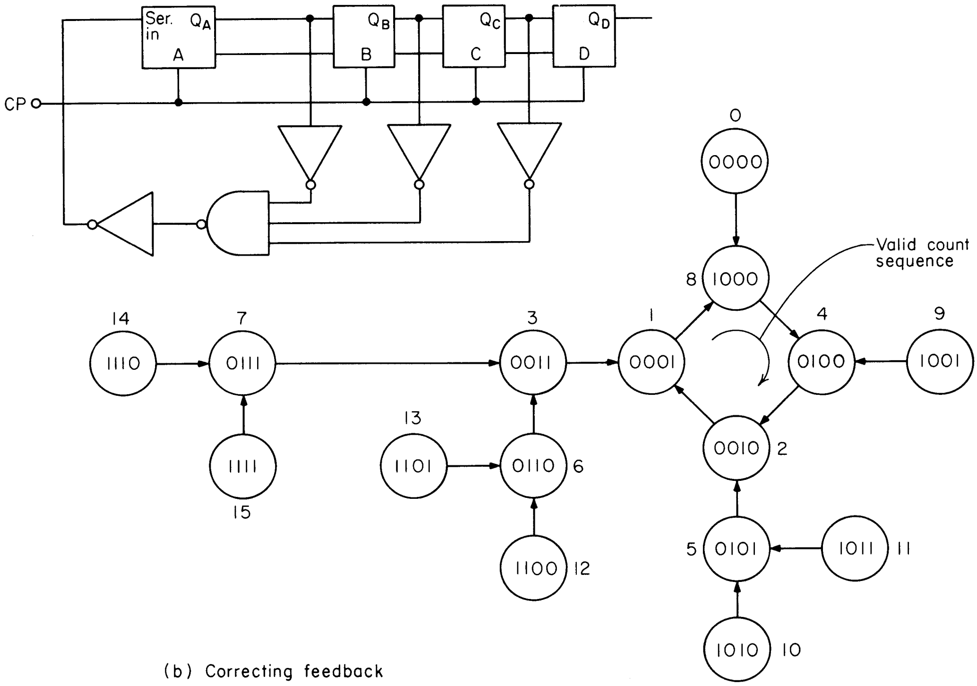
Clock pulse No.	Stage (F-F) No.				
	1	2	· · ·	N-1	N
0	1	0	· · ·	0	0
1	0	1	· · ·	0	0
2	0	0	· · ·	0	0
N-1	0	0	· · ·	0	1
N	1	0	· · ·	0	0

Cycle period = N

Fig. 11.10. Basic ring counter using SN54/7494 or SN54/7496.

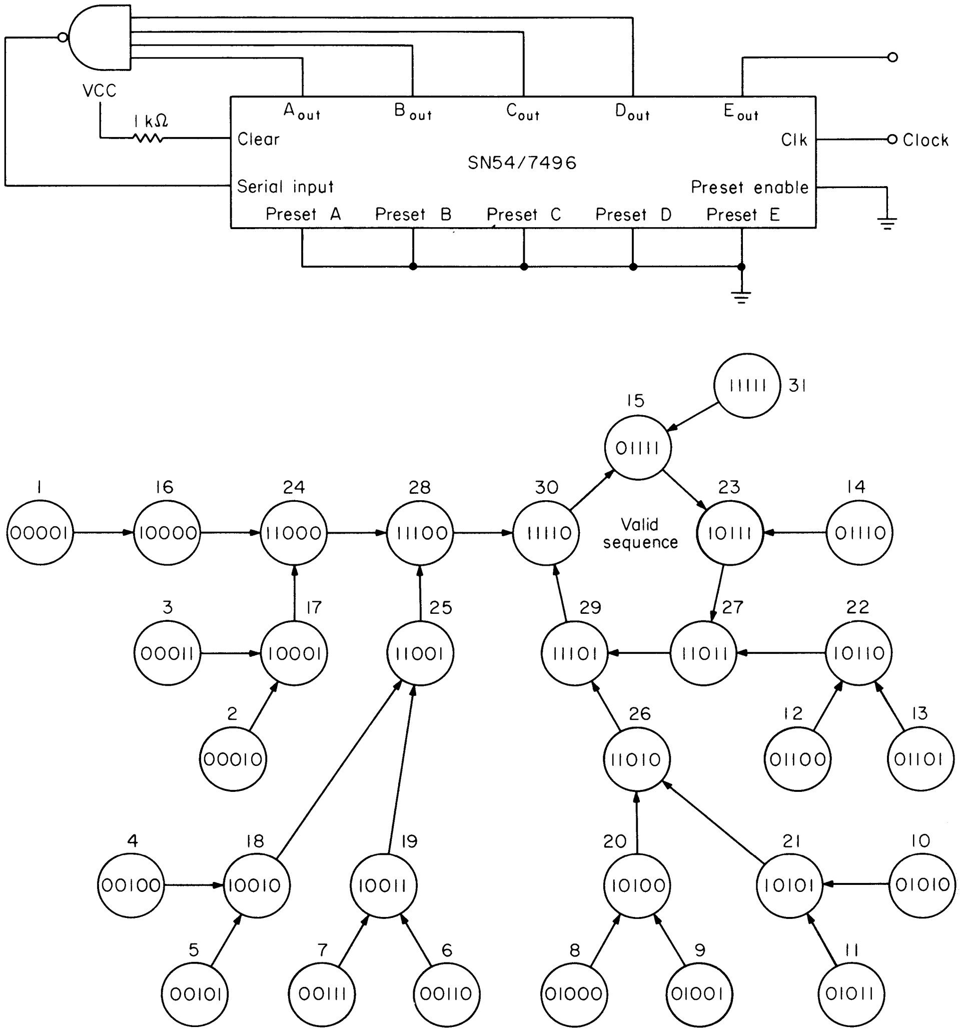

Ser. in
A
B
C
D
Q
Valid (desired) count sequence
1000
8
0001
1
0100
4
0010
2
0000
0
1111
15
1100
12
1001
9
0110
6
0011
3
1101
13
1011
11
1110
14
0111
7
0101
5
1010
10

(a) Basic feedback - showing all possible count sequences SN54/7495, L95, L98, L99


Ser. in
Q_A
A
Q_B
B
Q_C
C
Q_D
D
CP
0
0000
8
1000
Valid count sequence
14
1110
7
0111
3
0011
1
0001
4
0100
9
1001
1111
15
13
1101
0110
6
0010
2
1100
12
5
0101
1011
11
1010
10

(b) Correcting feedback

Fig. 11.11. 4-bit ring counter: illustration of self-start.


VCC
1 kΩ
A_{out}
B_{out}
C_{out}
D_{out}
E_{out}
Clear
Clk
Clock
SN54/7496
Serial input
Preset enable
Preset A
Preset B
Preset C
Preset D
Preset E
31 11111
15 01111
1 00001
16 10000
24 11000
28 11100
30 11110
Valid sequence
23 10111
14 01110
3 00011
17 10001
25 11001
29 11101
27 11011
22 10110
2 00010
26 11010
12 01100
13 01101
4 00100
18 10010
19 10011
20 10100
21 10101
10 01010
5 00101
7 00111
6 00110
8 01000
9 01001
11 01011

Note: A 0 is circulated for this case

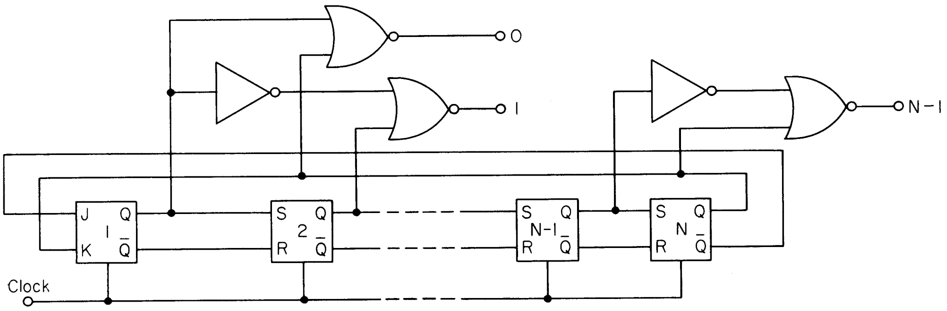
Fig. 11.12. 5-bit ring counter, self-starting.

Johnson Counters. The Johnson (also twisted-ring or Möbius) counter differs from the ring counter in that the complement of the last stage is fed back to the input. The result (Fig. 11.13) is a counter with $2N$ states (where N is the shift register bit length) and somewhat more complex decoding than the ring counter. A Johnson counter has $2^N - 2N$ unused states; thus for $N > 2$ there will be undesired stable sequences of states. The counter can enter an undesired sequence when power is first applied or it can enter because of noise. In the case of system turn-on, the counter can be cleared or preset or both, thus starting in a valid state. A 4-stage Johnson counter is analyzed in Fig. 11.14 with state diagrams showing all sequences. A Johnson counter can be made self-starting by using additional gating in the feedback to the J input. The counter is always self-starting (for N up to 25 stages) when feedback is provided from the last j stages of an N-stage shift register where

$$j > \frac{N}{3} \qquad j \text{ is an integer}$$

as shown in Fig. 11.14*b*.

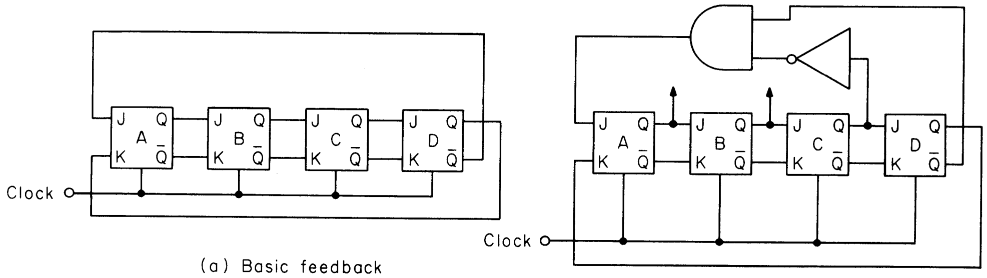
Up to this point only even-numbered cycle lengths ($2N$) have been considered. However, Johnson counters can also be made to have odd-numbered ($2N - 1$) cycle lengths by modification of the feedback. Figure 11.15 shows a technique that can be applied when the shift register has a double-rail J-K input available. In this



Clock pulse No.	Stage (F-F) No.						Decode logic (AND)
	1	2	3	. . .	N−1	N	
0	0	0	0		0	0	$\bar{1} \cdot \bar{N}$
1	1	0	0	. . .	0	0	$1 \cdot \bar{2}$
2	1	1	0		0	0	$2 \cdot \bar{3}$
3	1	1	1		0	0	$3 \cdot \bar{4}$
.				. . .			.
.				. . .			.
.							.
N − 1	1	1	1		1	0	$(N-1) \cdot \bar{N}$
N	1	1	1		1	1	$1 \cdot N$
N + 1	0	1	1		1	1	$\bar{1} \cdot 2$
N + 2	0	0	1		1	1	$\bar{2} \cdot 3$
N + 3	0	0	0		1	1	$\bar{3} \cdot 4$
.							
.				. . .			
.				. . .			
.							
2N − 1	0	0	0		0	1	$(\overline{N-1}) \cdot N$
2N	0	0	0		0	0	$\bar{1} \cdot \bar{N}$

Note: Count numbers 0, 1, and N − 1 are decoded above using NOR gates. All counts can be decoded in the same manner

Fig. 11.13. Basic Johnson counter.


J Q A K Q̄
J Q B K Q̄
J Q C K Q̄
J Q D K Q̄
Clock
Clock

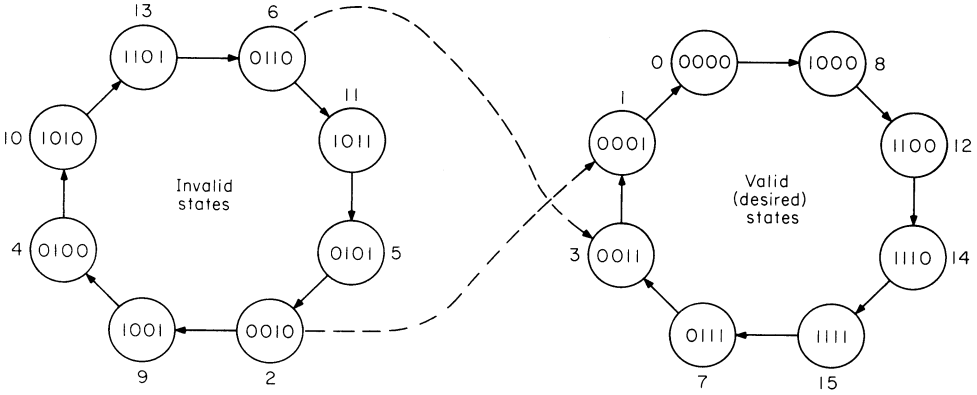
(a) Basic feedback

$\frac{N}{3} = \frac{4}{3} = 1.33$

$j > 1.33$

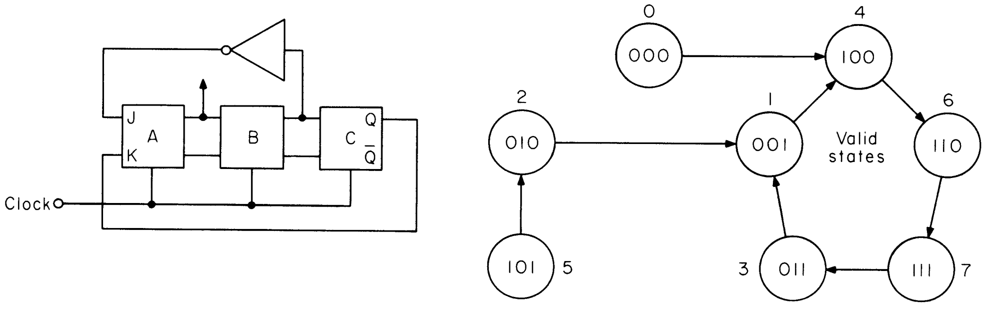
$j = 2$

(b) Modified feedback


13 1101
6 0110
11 1011
0101 5
2 0010
9 1001
4 0100
10 1010
Invalid states
0 0000
1000 8
1100 12
1110 14
1111 15
7 0111
3 0011
1 0001
Valid (desired) states

Dashed lines show result of modified feedback

Fig. 11.14. 4-stage Johnson counter: illustration of modified feedback for self-starting.


J A K
B
Q C Q̄
Clock
0 000
4 100
2 010
1 001
6 110
Valid states
101 5
3 011
111 7

Period = 2N − 1 = 5

Note: In this case, the counter is self-starting

Fig. 11.15. 3-bit Johnson counter with reduced cycle ($2N - 1$).

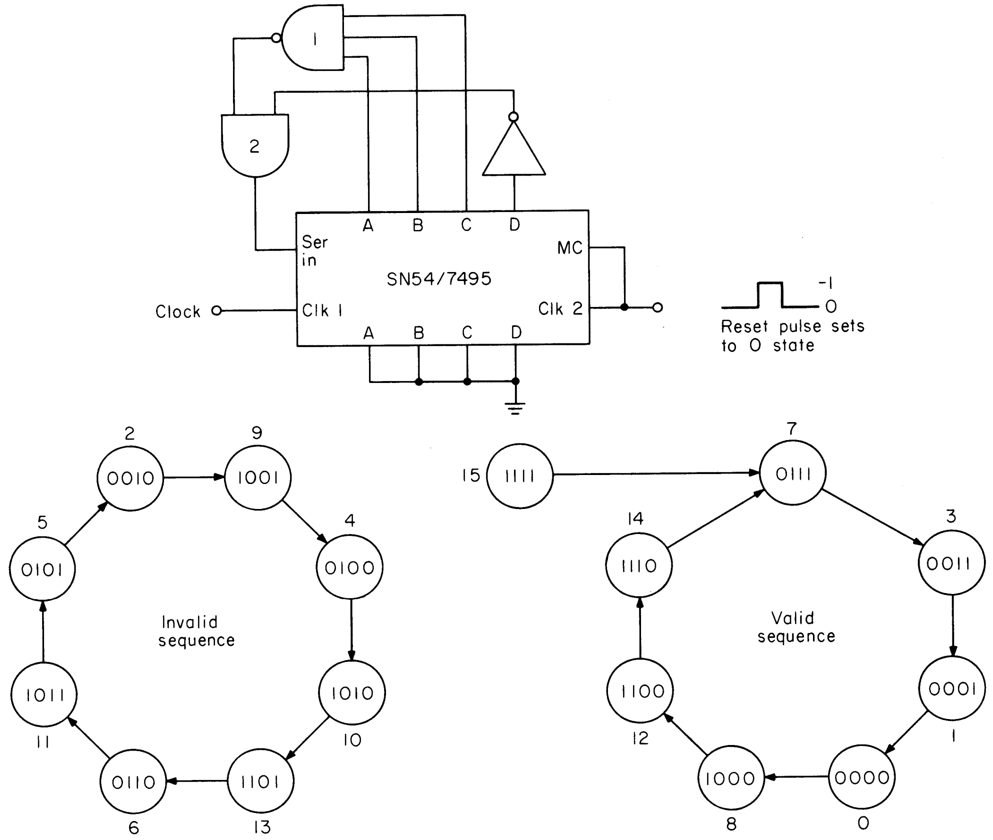

1
2
A B C D
Ser in
MC
SN54/7495
Clock
Clk 1
Clk 2
A B C D
-1
0
Reset pulse sets to 0 state
2 0010
9 1001
4 0100
10 1010
13 1101
6 0110
11 1011
5 0101
Invalid sequence
15 1111
7 0111
3 0011
1 0001
0 0000
8 1000
12 1100
14 1110
Valid sequence

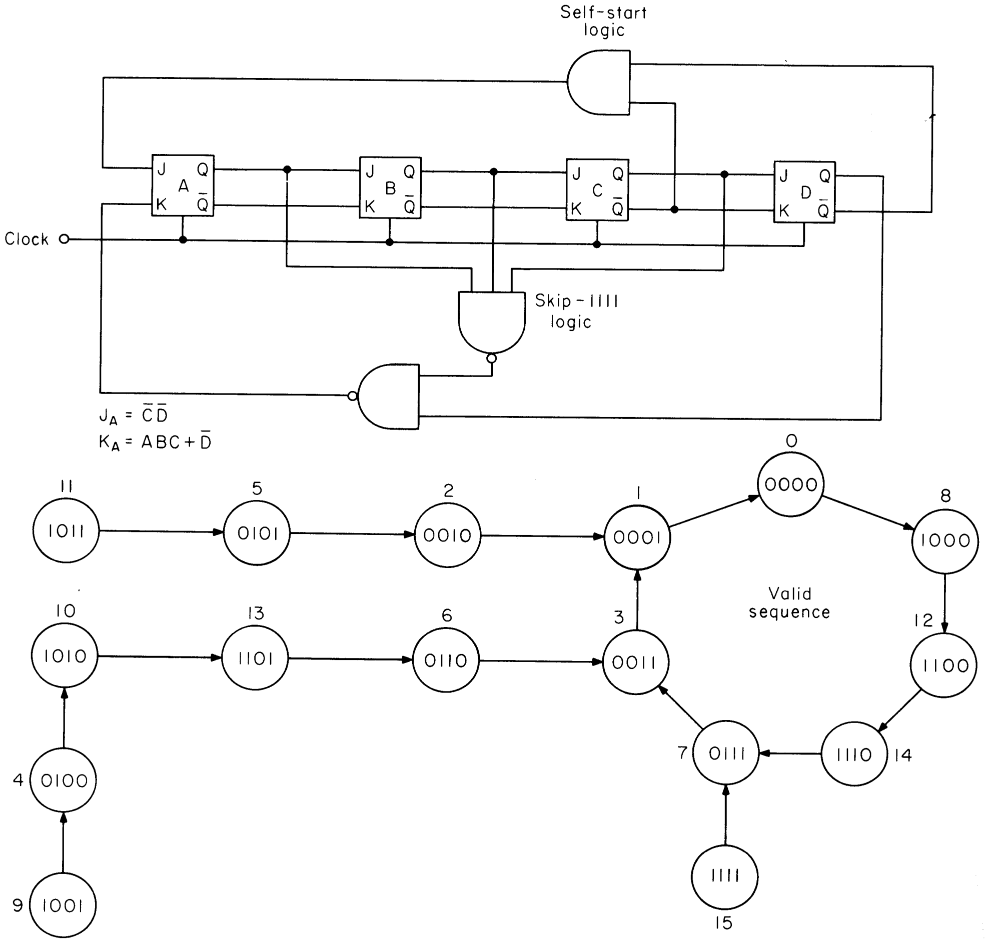
Fig. 11.16. 4-bit Johnson counter, odd cycle length, period = 7, non-self-starting.

method the all-0 state is skipped. For up to four stages, this method is self-starting without further modification.

Another technique for generating a $2N - 1$ cycle period skips the all-1 state. Figure 11.16 shows a Johnson counter of this type using SN54/7495. This counter is not self-starting, and so in this example it is reset to the 0000 state. Any other state can be programmed for reset at the *A*, *B*, *C*, and *D* inputs. This technique for generating a reduced cycle can be applied to achieve a shift register of any length, but the number of gate inputs for NAND gate 1 increases linearly with increase in shift register length.

Figure 11.17 illustrates the skip-all-1 state for an odd cycle length, but with the addition of self-starting logic. The self-starting logic follows essentially the same technique as presented in Fig. 11.14. The note of Fig. 11.17 indicates some of the possibilities for implementation using available shift registers or dual flip-flop packages or a combination of both.

Still another technique for generating an odd cycle length is shown in Fig. 11.18. This circuit is self-starting without further modification, and it differs from the

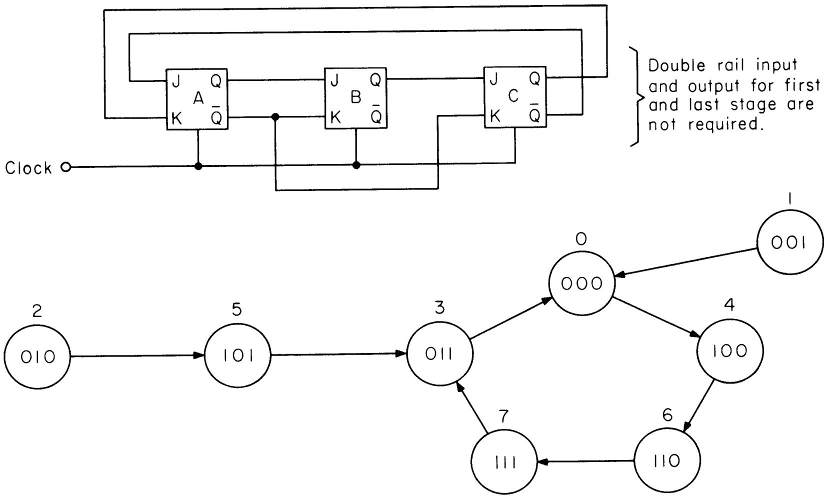

Self-start logic
J Q A K Q̄
J Q B K Q̄
J Q C K Q̄
J Q D K Q̄
Clock
Skip-1111 logic
J_A = C̄D̄
K_A = ABC + D̄
11 1011
5 0101
2 0010
1 0001
0 0000
8 1000
Valid sequence
10 1010
13 1101
6 0110
3 0011
12 1100
4 0100
7 0111
1110 14
9 1001
1111 15

Note: The shift register can be implemented using SN74107 dual J-K flip-flops, or an SN54/74L99 SR. For longer lengths in standard power logic, an SN54/74107 can be used for the first flip-flop in conjunction with the SN54/7495 or SN54/7496 SRs.

Fig. 11.17. 4-bit Johnson counter, odd cycle length, period = 7, self-starting.

previous odd-cycle Johnson counters in that it skips the 1 state (001). This technique can be applied to longer bit lengths with the same principles of operation.

Linear Shift Register Generator Counters. These counters are easier to build than synchronous binary counters, and they become economical for long cycle lengths. They are distinguished by the use of modulo-2 (exclusive-OR) feedback. These are synchronous counters and are divided into the subclasses of maximum-length (MLS) and non-maximum-length shift counters. A maximum-length shift is $2^N - 1$, where N is the number of shift register stages. Figure 11.19 includes a tabular listing of feedback terms for shift registers up to 12 stages in length. Since the exclusive-OR of 0s is 0, the all-0 state is a stable state from which the counter cannot exit without additional gating or a reset operation to a valid state. Because



Double rail input and output for first and last stage are not required.

Note: 1. The 001 state that belongs to the usual Johnson counter sequence is skipped

2. The shift register can be implemented for this bit length with the SN74107 dual J-K flip-flop, but for longer bit lengths any of the shift register packages can be used with an SN54/74107 flip-flop as the last one or two stages (depending on whether or not all flip-flop outputs are available for the package used)

Fig. 11.18. 3-bit Johnson counter, odd cycle length, period = 5, self-starting.

Number of stages in shift register	Feedback logic equation
3	$B \oplus C$
4	$C \oplus D$
5	$C \oplus E$
6	$E \oplus F$
7	$F \oplus G$
8	$D \oplus E \oplus F \oplus H$
9	$E \oplus I$
10	$G \oplus J$
11	$I \oplus K$
12	$F \oplus H \oplus K \oplus L$

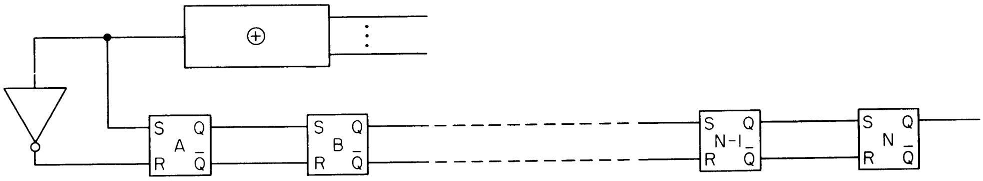
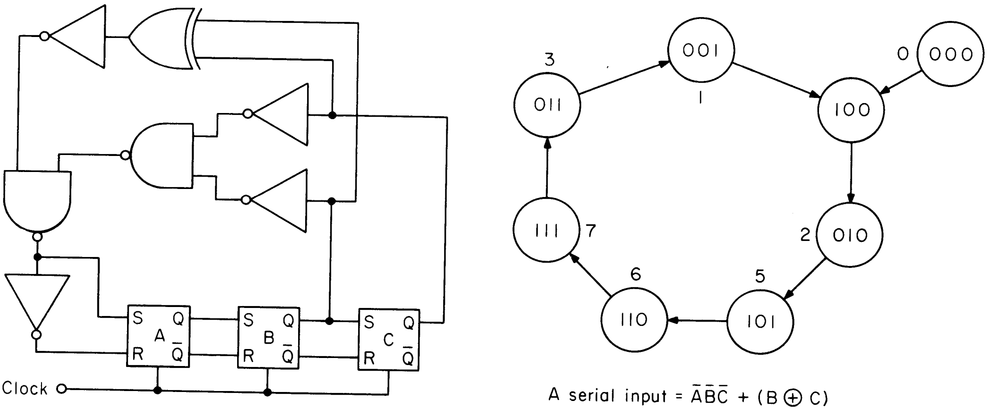


Fig. 11.19. Feedback logic for maximum length linear shift register generator counters.



A serial input $= \bar{A}\bar{B}\bar{C} + (B \oplus C)$
$= \bar{A}\bar{B}\bar{C} + B\bar{C} + \bar{B}C$
$= \bar{B}\bar{C} + B\bar{C} + \bar{B}C$

Fig. 11.20. 3-bit MLS shift register generator counter, with self-starting logic.

of this, these counters in their basic implementation are not self-starting. Figure 11.20 shows a 3-stage shift register generator counter with additional gating for exiting the 000 state.

There is a modification that allows this kind of counter to count to 2^N; this is one more than the usual definition of MLS, which is $2^N - 1$. The feedback is modified so that the exclusive-OR term is inhibited in the 0 . . . 001 state so that the next state becomes the 0 state (which is skipped in an MLS counter), and the following state is 2^{N-1}. For a 5-stage counter using this technique, the feedback logic equation is

$$F = (C + E) \cdot (\overline{\overline{ABCD}E}) + \overline{ABCDE}$$

Nonmaximum length refers to any cycle period less than $2^N - 1$. For non-MLS counters, a technique referred to as the *jump technique* is employed to skip over certain sequences of states, thereby reducing the total period. This is accomplished by additional feedback which detects the *jump state* and alters the feedback from its usual (MLS) value at the jump state. Figure 11.21 gives a table of jump terms for periods of from 4 through 63 (periods up to 2,047 are covered in the source†).

As an example using terms from Fig. 11.21, a counter with a period of 12 is shown in Fig. 11.22. The basic exclusive-OR term of the last two stages is used in conjunction with logic for the detection of the jump term. This circuit is not self-starting without further modification to enable an exit from the 0000 state.

The jump term can be found without using the table of Fig. 11.21. As an example, assume that a divide-by-8 counter is desired. This requires a 4-bit shift register that has an MLS period of 15. Figure 11.19 gives the basic feedback equation as the exclusive-OR of the last two stages. Using this information and starting with the 1111 state, the count sequence is written as shown in Fig. 11.23. The number of states to be jumped is found: $15 - 8 = 7$. The first 7 bits of the initial count

---

†*JPL Technical Report* 32-564.

sequence are omitted, and this sequence is written below the original sequence as shown in Fig. 11.23. The jump term is the 4-bit word in the upper row over the lower-row segment that reads 0001. The 4-bit words of the upper and middle row differ in only one bit position. Now to find the required state for the jump feedback, note that for this case, the basic modulo-2 feedback terms are *C* and *D*, and for the jump term indicated, the feedback would be a 1 level. The jump term must change the feedback to the opposite of its MLS state, and so a 0 level must occur as feedback for this jump term; this requires an inhibit of the modulo-2 feedback. For this example, a term is added to the feedback to make the counter self-starting; this means that the feedback level must be a 1 when in the 0000 state.

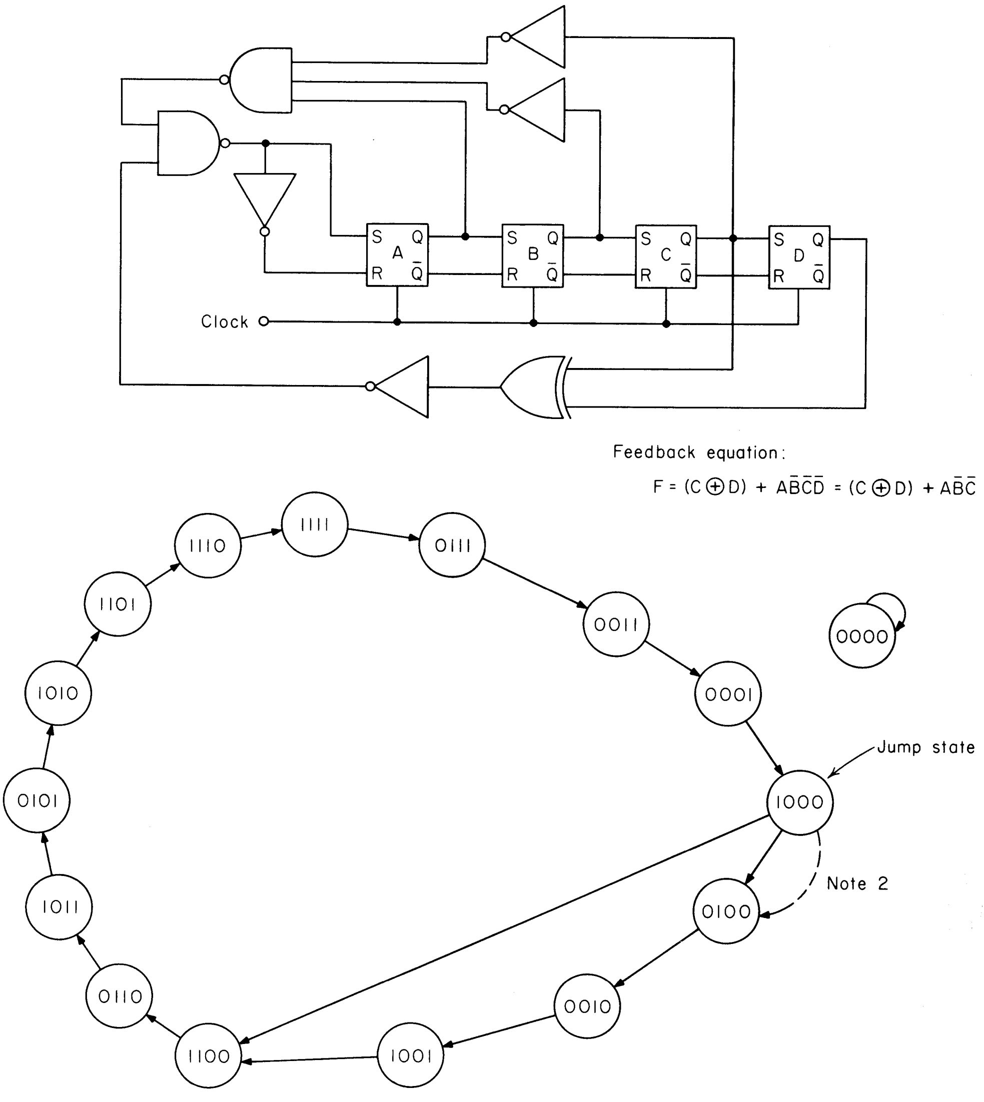
Shift Register Word Generators. These circuits are also known as shift register read-only memories. In designing the counters previously discussed, the overriding concern is with the number of states within a cycle period; but for word generators, the actual states and their sequence become the major concern.

Period	Stage						Note 1 Ser. in
	A	B	C	D	E	F	
4	0	1	1				1
5	1	0	0				1
6	1	1	0				0
7			MLS				
8	0	1	1	0			0
9	0	1	0	0			1
10	1	1	0	0			1
11	0	0	1	1			1
12	1	0	0	0			1
13	1	0	1	1			1
14	1	1	1	0			0
15			MLS				
16	1	1	0	1	0		1
17	1	0	0	0	1		0
18	0	0	0	1	1		0
19	0	1	1	0	1		1
20	1	1	1	0	0		0
21	1	0	1	0	1		1
22	0	1	1	1	1		1
23	0	1	0	0	1		0
24	0	0	1	1	0		0
25	1	0	0	1	0		1
26	0	0	1	0	1		1
27	1	0	1	1	0		0
28	0	1	0	0	0		1
29	0	1	0	1	1		0
30	1	1	1	1	0		0
31			MLS				
32	0	0	1	1	1	0	0
33	1	1	0	0	1	0	0
34	0	0	1	0	0	0	1
35	0	1	0	1	1	1	1
36	0	0	1	1	0	1	0
37	1	1	1	1	0	1	0
38	0	1	0	1	0	1	0
39	0	1	1	1	1	1	1
40	0	1	1	0	0	1	0
41	1	0	1	0	0	0	1
42	1	0	1	1	1	1	1
43	0	0	0	1	0	0	1
44	1	1	0	0	0	0	1
45	1	1	0	1	0	0	1
46	1	0	0	1	0	1	0
47	1	0	0	1	1	1	1
48	1	1	1	0	1	1	1
49	1	0	0	0	1	1	1
50	0	1	1	1	0	0	1
51	0	0	0	1	1	1	1
52	1	0	1	1	0	1	0
53	0	1	0	0	0	0	1
54	1	1	1	0	0	1	0
55	0	0	1	0	1	0	0
56	0	1	1	0	1	0	0
57	0	0	0	0	1	1	1
58	1	0	0	0	0	0	1
59	1	1	0	1	1	0	0
60	0	1	0	0	1	1	1
61	1	0	1	0	1	1	1
62	1	1	1	1	1	0	0
63				MLS			

Source: JPL Technical Report 32-564

Note 1: Logic value for serial input to shift register upon detection of jump term

Fig. 11.21. Jump terms for non-MLS shift register generator counters.


S Q A R Q̄
S Q B R Q̄
S Q C R Q̄
S Q D R Q̄
Clock
1111
1110
0111
1101
0011
0000
1010
0001
Jump state
0101
1000
Note 2
1011
0100
0110
0010
1100
1001

Feedback equation:

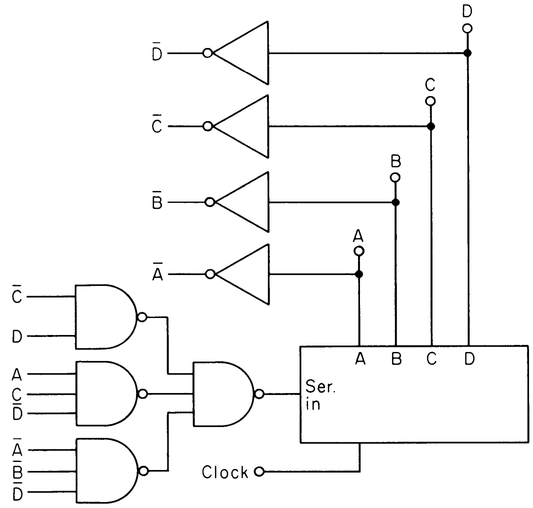
$$F = (C \oplus D) + A\bar{B}\bar{C}\bar{D} = (C \oplus D) + A\bar{B}\bar{C}$$

Note:

1. An SN7495, SN74L95, or SN74L99 is appropriate for implementation of this counter
2. Dotted line shows path without jump-term feedback

Fig. 11.22. Non-MLS shift register generator counter with period = 12, non-self-starting.

The design of word generators is illustrated by the following example: generate the decimal numbers 6,7, 8, 9 BCD equivalents cyclically, using a minimum-stage shift register. The required sequence, starting first with the rightmost digit and least significant bit, is shown in Fig. 11.24. Since the sequence is 16 bits in length, the minimum circuit to consider is a 4-stage shift register. But inspection of the sequence shows that the shift register must have more than four stages to prevent pattern


D
C
B
A
$\bar{D}$
$\bar{C}$
$\bar{B}$
$\bar{A}$
$\bar{C}$
D
A
C
$\bar{D}$
$\bar{A}$
$\bar{B}$
$\bar{D}$
A B C D
Ser. in
Clock

$$F = \underbrace{(C \oplus D)}_{\substack{\text{MIS}\\\text{feedback}}} \cdot \underbrace{(\overline{\bar{A}BC\bar{D}})}_{\substack{\text{Jump}\\\text{term}\\\text{(inhibit)}}} + \underbrace{\bar{A}\bar{B}\bar{C}\bar{D}}_{\substack{\text{Exit}\\\text{0000}\\\text{state}}} = \bar{C}D + AC\bar{D} + \bar{A}\bar{B}\bar{D}$$

Note: An SN7495, SN74L95 or SN74L99 is appropriate for implementation of this counter

																						A	B	C	D
Original count sequence	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1	0	0	0	1	1	1	1
												Jump term						Omit first 7 bits						
Original count sequence	0	1	0	0	0	1	1	1	1	0	1	$\boxed{0}$	$\boxed{1}$	$\boxed{1}$	$\boxed{0}$	0	1	0	0	0	1	1	1	1
Shifted right by 7 bits				0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0	0	1
Modulo-2 sum (exclusive-OR of each column)				0	1	0	1	1	0	0	1	0	0	0	1	1	1	1	0	1	0	1	1	0

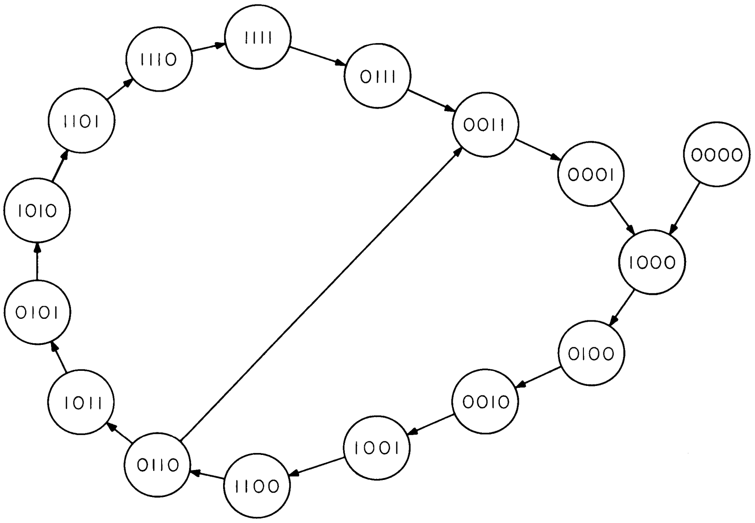
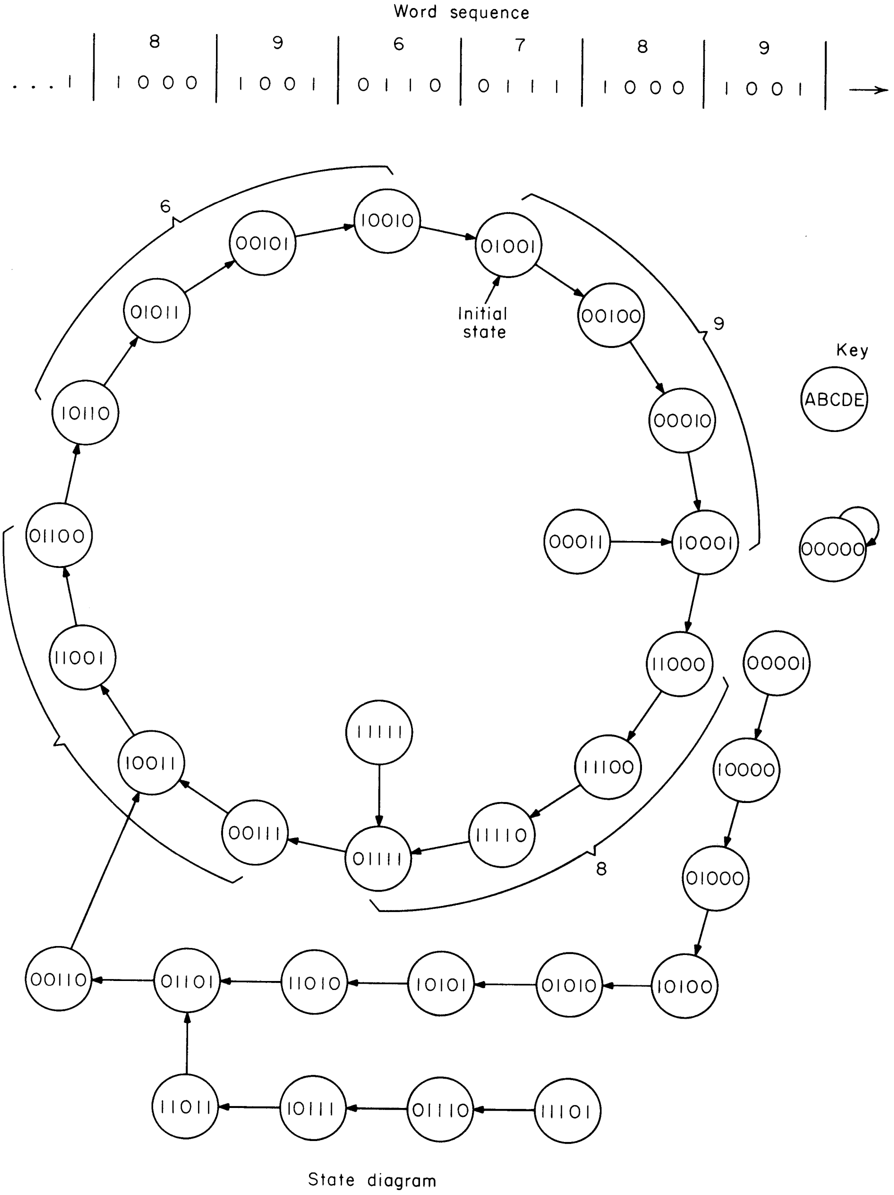

1111
0111
0011
0001
1000
0100
0010
1001
1100
0110
1011
0101
1010
1101
1110
0000

Fig. 11.23. Non-MLS shift register generator counter with period = 8, self-starting.

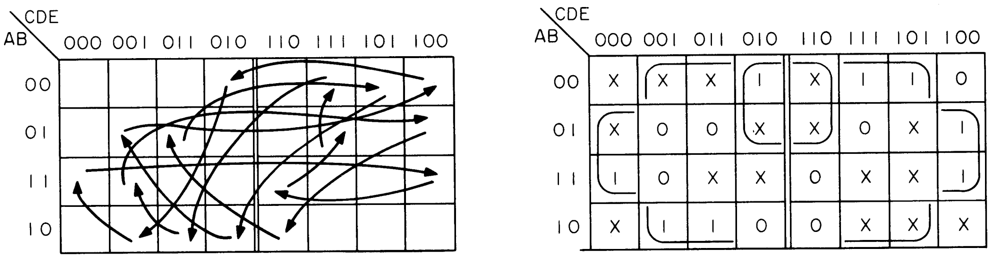

Word sequence

State diagram

. . . 1
8 1000
9 1001
6 0110
7 0111
8 1000
9 1001
6
9
8
Key
ABCDE
Initial state
00101
10010
01001
00100
00010
10001
00011
11000
11100
11110
01111
11111
00111
10011
11001
01100
10110
01011
00000
00001
10000
01000
10100
01010
10101
11010
01101
00110
11011
10111
01110
11101

Fig. 11.24. Shift register word generator.

repetition. Notice that the first four bits, 1001, appear again in the sequence as the last two bits of BCD 7 and the first two bits of BCD 6. Therefore, a 4-stage shift register does not have a unique present state for each required next state. A sequence using a 5-stage shift register will create the 16 unique states—one for every bit generated. A state diagram is now generated (for convenience) by taking each 5-bit group of the sequence going from right to left. A transition map is then

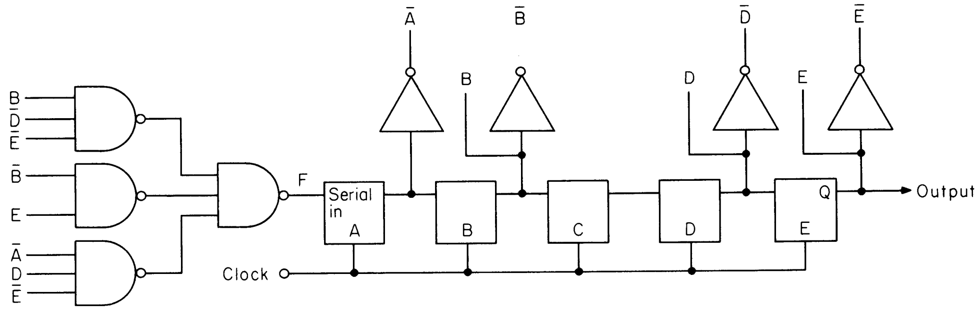


AB \ CDE	000	001	011	010	110	111	101	100
00	X	X	X	1	X	1	1	0
01	X	0	0	X	X	0	X	1
11	1	0	X	X	0	X	X	1
10	X	1	1	0	0	X	X	X

$F = B\bar{D}\bar{E} + \bar{B}E + \bar{A}D\bar{E}$

Transition map

Next-state map



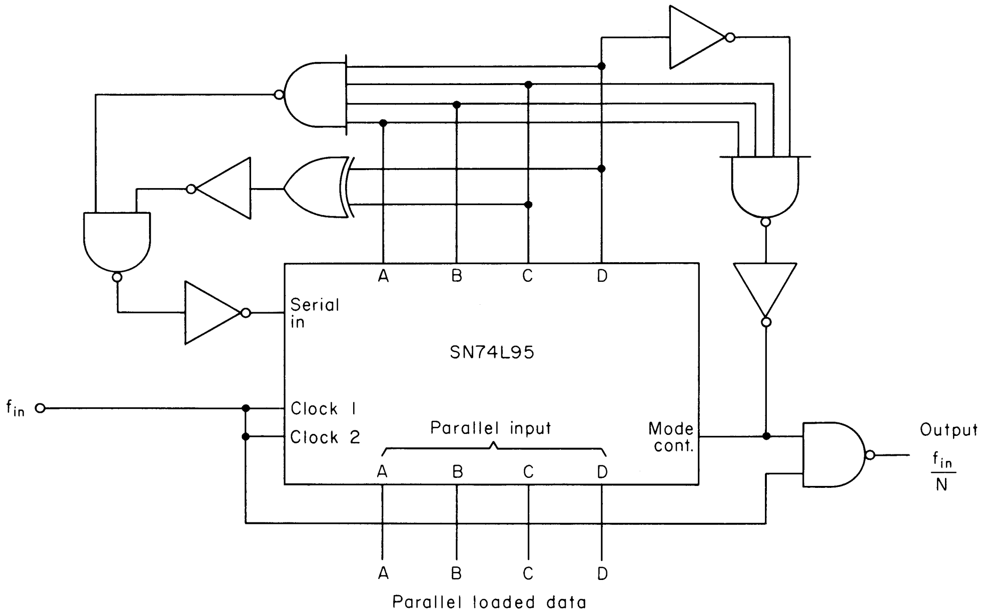
Note: The SN54/7496 are appropriate shift registers for implementation of this circuit

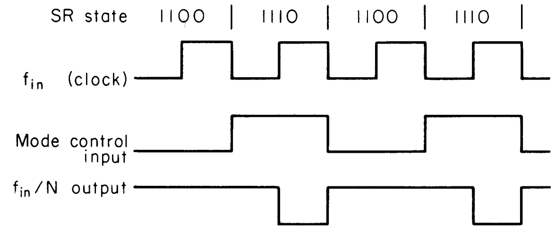
Figure 11.24. (*Continued*)

constructed, using the state diagram or the sequence. It is convenient to label the map so that $A = 0$ (A is the first stage of the shift register) identifies the two upper rows and $A = 1$ identifies the two lower rows. The unused cells on the map contain X-type "don't cares," but these are omitted from the transition map to avoid clutter; they are entered on the next-state map. The next-state map is generated from the transition map by taking each active cell and finding the value of A for the next state. This value is entered in the cell under consideration in the next-state map. When this is completed, the usual minimization techniques can be applied. The resulting minimized function is F. Now using this function, the remaining unused states and their transitions are added to the original state diagram. Except for the all-0 state, the shift register generator is self-starting. Exit from the all-0 state can be accomplished by detecting the all-0 state and combining this signal with the existing feedback logic.

11.3 OTHER SHIFT REGISTER APPLICATIONS

Shift Register Modulo-N ($1 \leq N \leq 16$) Frequency Divider. This circuit is especially useful in low-power applications because there are no programmable counters in the 54/74L low-power TTL series. The function performed here can be more easily and usefully implemented by using the SN54/74192 or SN54/74193 when


A
B
C
D
Serial in
SN74L95
f_{in}
Clock 1
Clock 2
Parallel input
A
B
C
D
Mode cont.
Output
$\frac{f_{in}}{N}$
A
B
C
D
Parallel loaded data


SR state
1100
1110
1100
1110
f_{in} (clock)
Mode control input
f_{in}/N output

Timing diagram – for divide-by-2; N = 2; load data = 1100

Transition and code conversion table
(N to load data)

N	Load data and state			
	A	B	C	D
16	1	1	1	1
15	0	1	1	1
14	1	0	1	1
13	1	1	0	1
12	0	1	1	0
11	0	0	1	1
10	1	0	0	1
9	0	1	0	0
8	1	0	1	0
7	0	1	0	1
6	0	0	1	0
5	0	0	0	1
4	0	0	0	0
3	1	0	0	0
2	1	1	0	0
1	1	1	1	0*

* Parallel load at this state

Fig. 11.25. Shift register variable modulus frequency divider ($N = 1$ through 16).

power dissipation is not a consideration. These ICs are described in Chapter 10, "Counters." The basic circuit is an MLS shift register generator counter previously described, except that the complement of the exclusive-OR is used for the basic feedback. The reason for this is that fewer inverters are required to detect the *end or parallel-load* state and the *latch-up* state (the latch-up state becomes 1111 rather than 0000). The transition and code conversion table of Fig. 11.25 shows the transition of the shift register state: the transition sequence is from top to bottom. The table also associates the number (*N*), by which the frequency is divided, to the required data for the SN54/74L95 parallel-load inputs. (This necessity for a code conversion is avoided when a modulo-*N* frequency divider is implemented using a programmable down counter.) For $N = 1$ the shift register is always in state 1110, and f_{in} is gated through.

Shift Registers for Number Conversion Applications. Circuits for serial binary to parallel BCD and serial BCD to serial binary conversion using the SN54/7495 are shown in Chapter 9, "Arithmetic Elements."

12

Other Applications

This chapter presents various applications of TTL integrated circuits in the hope that they will solve specific user problems, or at least generate new concepts for the design of related circuits.

12.1 A SIMPLE BINARY MULTIPLIER

The many methods of digital multiplication range from simple multiple addition systems to high-speed static multipliers. The former are economical but very slow; the latter can be very fast, but the complexity increases roughly as the square of the number of bits to be multiplied. The dynamic system described here is a good compromise between speed and simplicity, and it is suitable for desk calculators. It is equivalent to a pencil-and-paper method as shown below for two 4-bit numbers X and Y.

$$\begin{array}{r} Y = 1011 \\ X = \underline{1001} \\ 1011 \\ 0000 \\ 0000 \\ \underline{1011} \\ 1100011 \end{array}$$

Here the product is obtained by multiplying Y successively by the separate digits of X and summing the partial products. Shifting Y to the left by n bits is, of course, equivalent to multiplying by 2^n.

SN7495N is a 4-bit shift register which can be changed to a 4-bit parallel-in/parallel-out register by altering the logic level on a mode control. It is therefore particularly suitable for the add and shift operation used here as well as for a shift-left/shift-right function or for serial-to-parallel or parallel-to-series conversion.

Refer to the multiplier logic diagram Fig. 12.1. Initially, the reset input level is a logical 1, which clears the flip-flops $FF1$, $FF2$, and $FF3$ and changes the mode

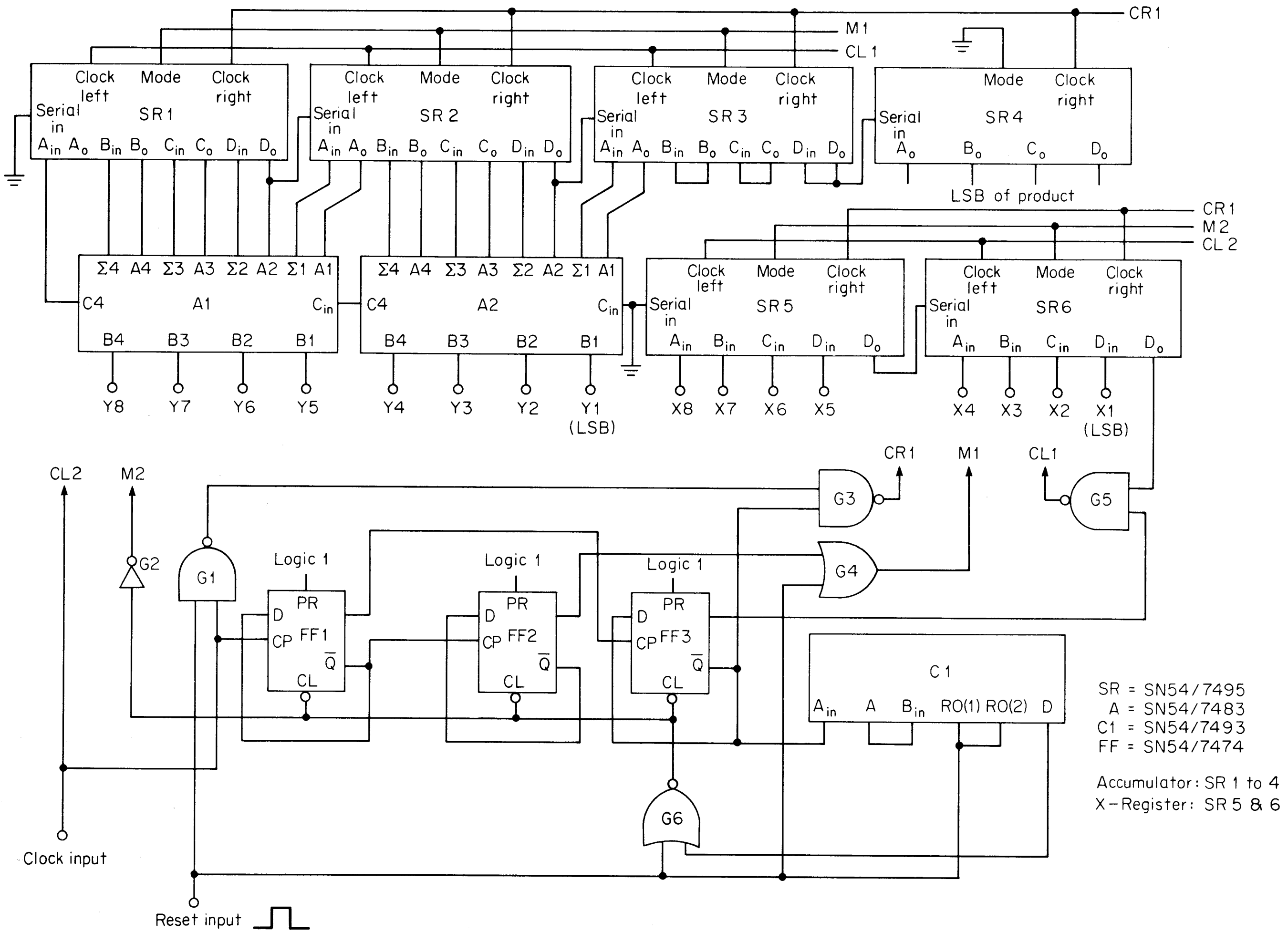

CR1
M1
CL1
Clock left
Mode
Clock right
Serial in
SR1
SR2
SR3
SR4
A_{in} A_o B_{in} B_o C_{in} C_o D_{in} D_o
LSB of product
CR1
M2
CL2
Σ4 A4 Σ3 A3 Σ2 A2 Σ1 A1
C4
A1
A2
C_{in}
B4 B3 B2 B1
Y8 Y7 Y6 Y5
Y4 Y3 Y2 Y1 (LSB)
SR5
SR6
X8 X7 X6 X5
X4 X3 X2 X1 (LSB)
CR1
M1
CL1
CL2
M2
G1
G2
G3
G4
G5
G6
Logic 1
PR
D
CP
CL
$\overline{Q}$
FF1
FF2
FF3
C1
A_{in} A B_{in} RO(1) RO(2) D
Clock input
Reset input
SR = SN54/7495
A = SN54/7483
C1 = SN54/7493
FF = SN54/7474
Accumulator: SR 1 to 4
X-Register: SR 5 & 6

Fig. 12.1. Multiplier logic diagram.

$M2$ of shift registers $SR5$ and $SR6$ to parallel entry. The first pulse to arrive at the clock left ($CL2$) input during this standby period will enter the multiplier X into $SR5$ and $SR6$. When the clear inputs on networks on $FF2$ and $FF3$ rise to a logical 1 after the reset period, the multiplication starts. $FF1$ is used to ensure that the clock inputs to $FF2$ and $FF3$ start in the correct phase.

A typical multiplication sequence is shown in Fig. 12.2 for an X input of 10101101. The first positive edge from output $FF1$ causes $FF3$ to rise to a logical 1, which enables gate $G5$. The D output of $SR6$ is taken to the other input of gate $G5$, and since the least significant bit of X, X_1, is a logical 1, a negative edge is applied to the clock left input $CL1$ of the accumulator networks $SR1$, $SR2$, and $SR3$. The mode control $M1$ is a 1 at this point so that the multiplicand Y is entered into the accumulator via the SN7483N 4-bit adders $A1$ and $A2$. The D_0 output of $SR6$ inhibits or enables gate $G5$ and hence controls the parallel entry into networks $SR1$, $SR2$, and $SR3$; this is equivalent to multiplying Y by X_1.

It is now necessary to shift the next bit of X, X_2, to the D_0 output and to steer the next addition into a higher-power position in the accumulator. This could be done by shifting Y to the left. However, this approach would need extra registers and adders, and so it is more economical to shift the contents of the accumulator to the right. After each addition step, therefore, the mode line $M1$ is changed to 0, taking care to avoid generating an extra clock pulse when the mode level changes. This means that the clock right input ($CR1$) must be a 0 for a negative mode edge, and clock left ($CL1$) must be a 0 for a positive mode edge. Flip-flop $FF2$ generates a second phase so that the mode control can be changed without the risk of an incorrect or ambiguous state at the clock inputs.

The accumulator and the X register are now clocked by $CR1$ one position to the right by a negative edge from output $\overline{FF3}$ via gate $G3$. The mode line $M1$ then returns to a 1 as output $FF2$ falls to 0, and the *add-and-shift* cycle is complete. After eight additions and seven shifts the XY appears in the accumulator with the least significant bit in the D_0 output of $SR4$.

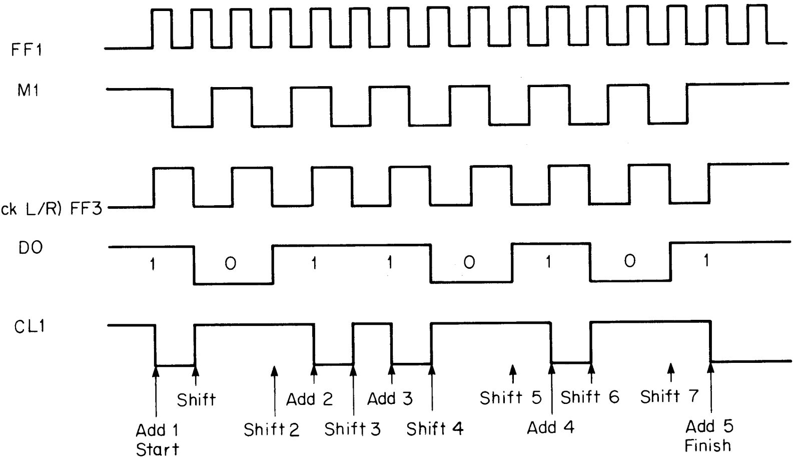


Fig. 12.2. Typical multiplication sequence for $X = 10101101$.

To stop the sequence at this stage, a 4-bit binary counter *C*1 (SN7493N) is added to the circuit. This is clocked by each shift-right pulse from the *FF*3 output. On the eighth pulse the *D* output of *C*1 rises to a logical 1, and the clear inputs of *FF*1 and *FF*2 are enabled via gate *G*6. While the reset line is at a 1, the mode line *M*1 drops to a 0 owing to gate *G*4, and gates *G*1 and *G*3 are enabled. The main clock now shifts 0s into the accumulator from the serial input on *SR*1 at four times the normal shift rate, after which the reset input is returned to 0 and the whole multiplication process repeats.

Figure 12.3 shows the approximate maximum speed of operation for one add-and-shift cycle based on typical propagation delays. This indicates a delay of 165 ns per step or about 5.5 μs for a 32- by 32-bit product.

The circuit shown was built up on a standard patchboard and ran satisfactorily at a quarter of this speed. However, since no special precautions were taken with grounding, layout, or decoupling, it is anticipated that something approaching the theoretical maximum speed should be practicable. Maximum operation speed would probably entail the use of a high-speed TTL flip-flop (Series 74H0 for *FF*1) to achieve the required toggle frequency.

The circuit is easily extended by adding one SN54/7495N and one SN54/7483N

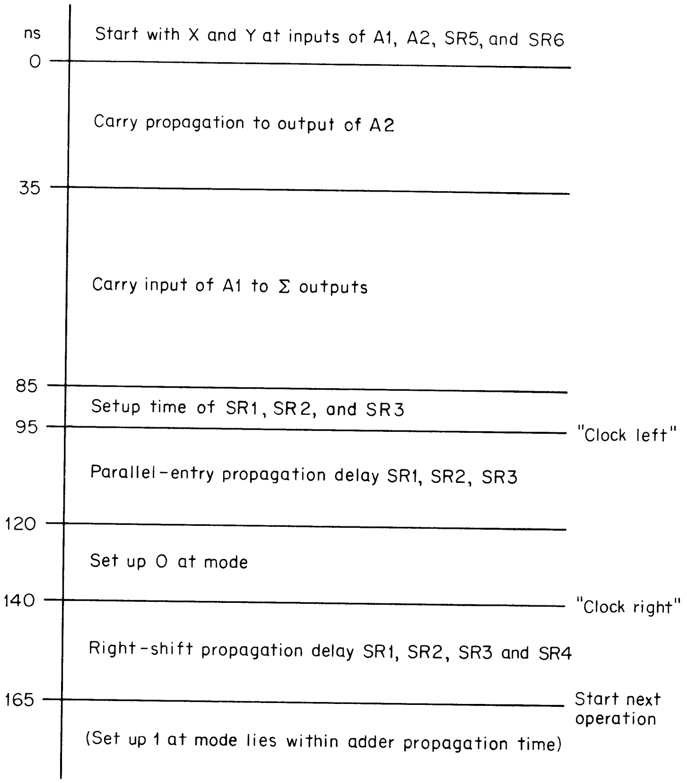

ns
Start with X and Y at inputs of A1, A2, SR5, and SR6
0
Carry propagation to output of A2
35
Carry input of A1 to Σ outputs
85
Setup time of SR1, SR2, and SR3
95
"Clock left"
Parallel-entry propagation delay SR1, SR2, SR3
120
Set up 0 at mode
140
"Clock right"
Right-shift propagation delay SR1, SR2, SR3 and SR4
165
Start next operation
(Set up 1 at mode lies within adder propagation time)

Fig. 12.3. Typical add-and-shift operating times.

for every 4-bit increase in the multiplicand Y, or by adding two SN54/7495Ns for every 4-bit increase in X. It is also possible to rearrange the system to form a binary divider.

12.2 12-HOUR DIGITAL CLOCK

A simple 12-hour digital clock is shown in Fig. 12.4. The system is synchronized with the 60-Hz power-line frequency. The SN74121 one-shot is triggered at its Schmitt input by 60-Hz pulses from the bridge rectifier. The one-shot pulse width is set for 15 ms (about 90 percent of the 16.7-ns period of the line frequency) so that power-line noise is inhibited for 90 percent of the period. An indicator for AM and PM, using small incandescent lamps, is shown. If this indicator is not desired, then the connection of flip-flop A to the tens-of-seconds counter and the connection of the NAND gate to its input may be deleted.

There are five switches provided for *setting* the clock. The procedure for setting is: (1) When the desired indication for seconds appears, switch $S1$ is placed in the *set* position. This stops the clock by inhibiting the one-shot clock-pulse generator. (2) Switch $S2$ is placed in the set position. Switch $S5$ (a spring-loaded push button) is then operated by pushing and releasing until the desired number is obtained for the minutes unit digit. The counter transition occurs when the push button is released. Switch $S2$ is now returned to the *normal* position. If a toggle switch is substituted for the push button $S5$, the switch should always be left in the position which causes the counter to step (the grounded terminal of the switch) before $S2$ is returned to the normal position. (3) The procedure of step 2 is repeated for $S3$ (tens-of-minutes) and then $S4$ (hours). $S4$ is also used to set the AM and PM indicator. Note that if the hours counters are set past 12, the tens-of-minutes counter will reset to 0. (4) With all digits set and $S2$, $S3$, and $S4$ in the normal position, $S1$ is set to the normal position and the clock begins to run. The unregulated +5-V power supply is adequate where good-quality line regulation exists. For line regulation noticeably poorer than ±5 percent, a regulated power supply for +5 V will be necessary. Burroughs Nixie® tube displays are indicated; for other possibilities, refer to Chapter 8, "Decoders." Note that the outputs of the SN74141N devices that are connected to the BCD outputs of the SN7492 counters are rearranged for connection to the Nixie tube displays. If 7-segment displays are used, the *A-B-C* inputs should be used to drive the decoder/driver. An extra flip-flop package will then be necessary.

12.3 SERIAL GRAY CODE TO BINARY CONVERSION

A simple technique for converting the special case of a reflected Gray code is shown in Fig. 12.5. Note that if the most significant bit (MSB) is disregarded, the numbers 8 through 15 will be found to be mirror images of numbers 0 through 7.

It can be shown that if G_j is the jth bit in the reflected Gray code and B_j is the corresponding bit in the equivalent binary number, then

$$B_0 = G_0$$

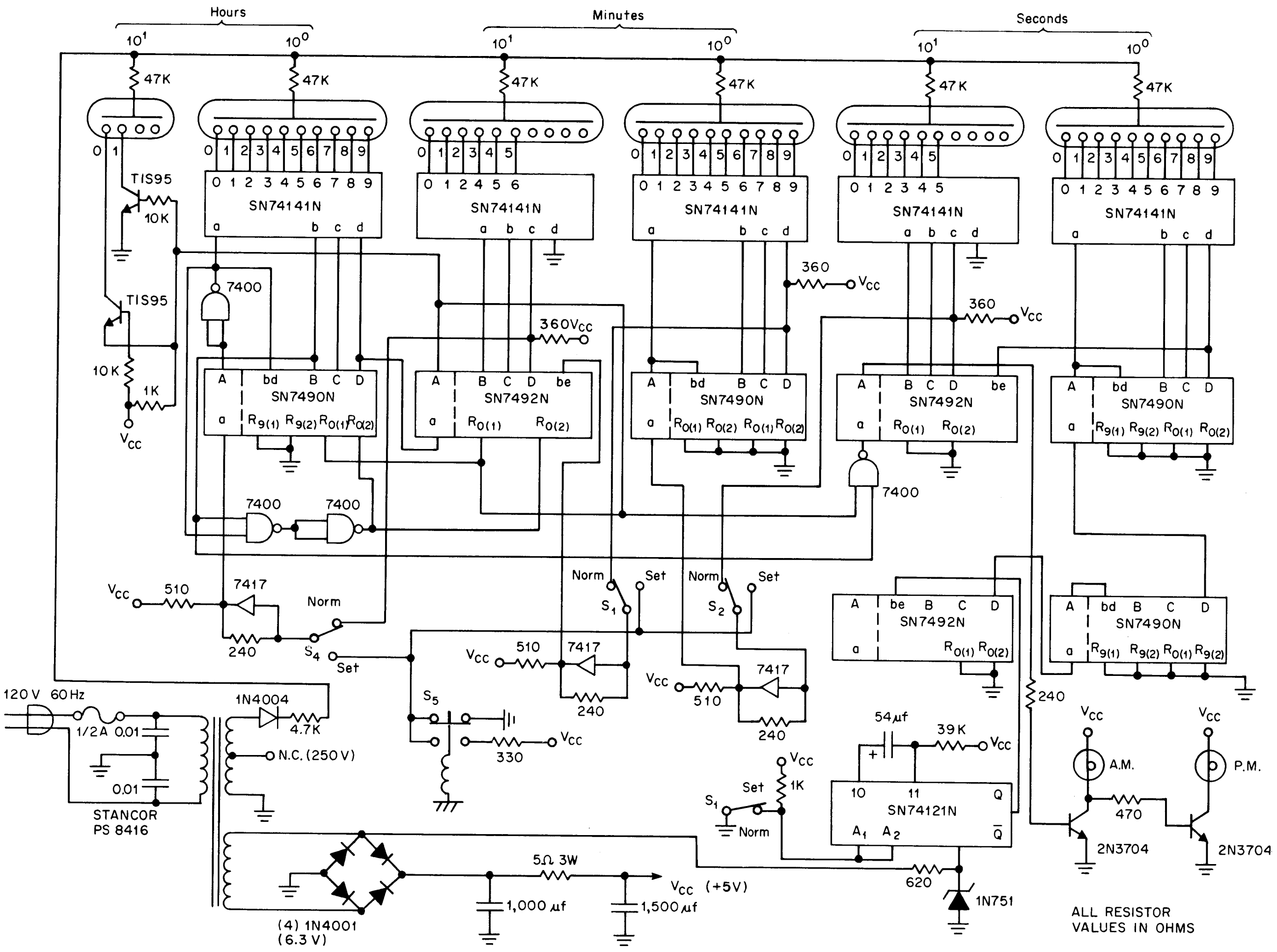

Hours
Minutes
Seconds
10^1
10^0
47K
SN74141N
SN7490N
SN7492N
SN74121N
A
a
bd
be
B C D
b c d
R9(1) R9(2) R0(1) R0(2)
7400
7417
TIS95
10K
1K
Vcc
360
510
240
Norm
Set
S1
S2
S4
S5
330
120 V 60Hz
1/2A
0.01
STANCOR PS 8416
1N4004
4.7K
N.C. (250 V)
(4) 1N4001 (6.3 V)
5Ω 3W
1,000 μf
1,500 μf
Vcc (+5V)
54μf
39K
10
11
Q
Q̄
A1 A2
620
1N751
A.M.
P.M.
470
2N3704
ALL RESISTOR VALUES IN OHMS

Fig. 12.4. 12-hour digital clock.

	G_0	G_1	G_2	G_3
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

Fig. 12.5. Conversion of reflected Gray code to binary.

where B_0 and G_0 are the MSBs and

$$B_j = B_{j-1} \oplus G_j$$

where (word length) $-1 \geq j > 0$.

These equations are easily implemented with a single *J-K* flip-flop. Figure 12.6 shows the connections for the converter. The same technique may be applied for the case where the data are in parallel form. Figure 12.7 illustrates this idea. A shift register is used to make a parallel-to-serial conversion. Note that the converted binary word is restored in the shift register.

As an example, consider the conversion of the Gray representation of 9 (1101) to binary (1001). Figure 12.8 shows the sequence of operations for a circuit such as Fig. 12.7 and is also representative of the serial-to-serial conversion.

12.4 MODULO-360 ADDER

The most obvious use for a modulo-360 adder is the addition of compass headings in navigational instrumentation or other applications where noncumulative addition is desired. (See Fig. 12.9.) The units adder is a conventional BCD adder (modulo-10) which upon reaching sums of 10 and above produces a carry and

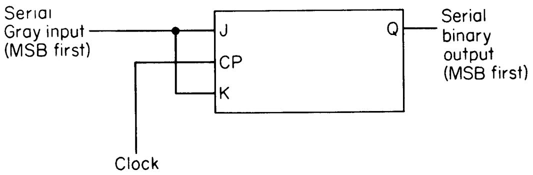

Serial Gray input (MSB first)
J
CP
K
Q
Serial binary output (MSB first)
Clock

Fig. 12.6. Serial Gray to serial binary converter.

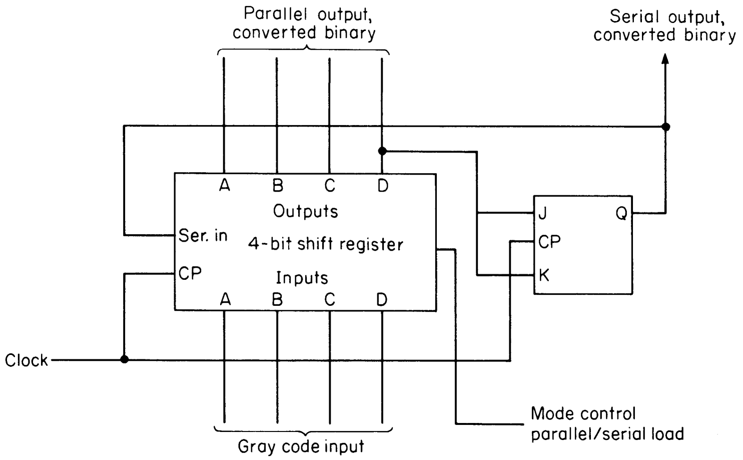
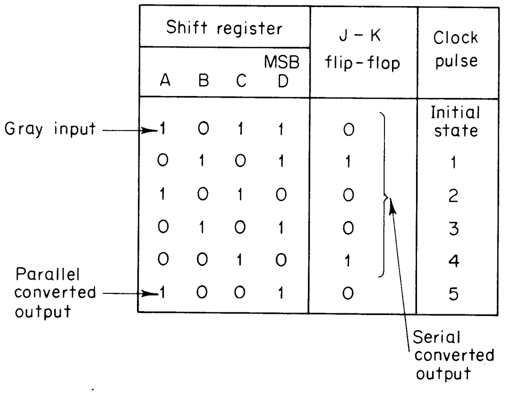

Parallel output,
converted binary
Serial output,
converted binary
A
B
C
D
Outputs
Ser. in
4-bit shift register
CP
Inputs
A
B
C
D
J
Q
CP
K
Clock
Mode control
parallel/serial load
Gray code input

Fig. 12.7. Parallel Gray code to parallel or serial binary converter.

subtracts 10 from the lower adder (*Z*2). The subtraction is implemented by the addition of 0110_2 (6); this is the 2's complement of the BCD representation of 10. Additional modulo-10 adders can be placed to the right of the decimal point for tenths, hundredths, etc.

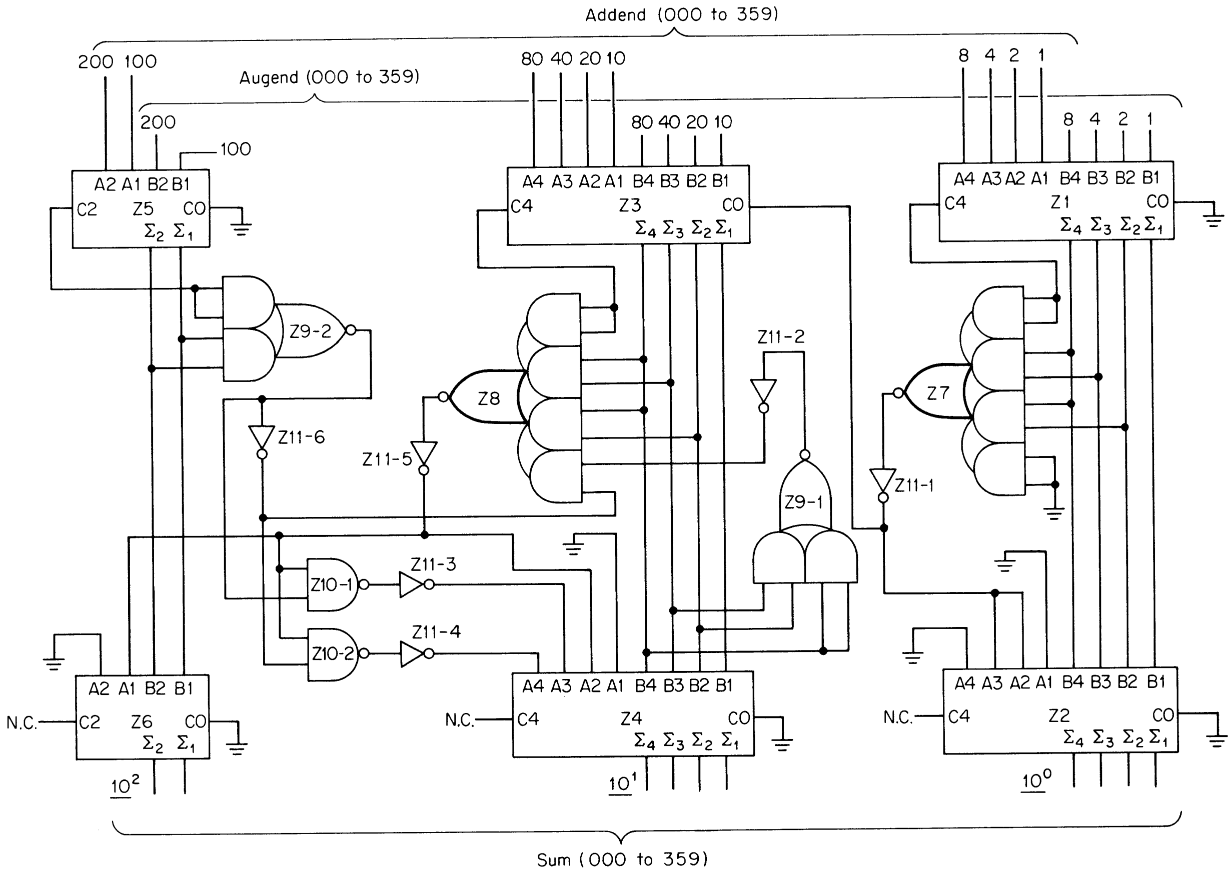
The tens decade contains additional logic because it is a dual-modulus adder. If the sum of the hundreds decade (disregarding a carry into the hundreds decade) is less than or equal to 2, then the tens decade is a modulo-10 adder; if the sum is 3 or greater, then the tens decade is a modulo-6 adder. This adder is implemented by using the same scheme as used in a BCD modulo-10 adder to detect numbers greater than or equal to 10 (performed by *Z*8), and 10 is effectively subtracted when required, as previously described for the units decade. For modulo-6 addition, *Z*9-1 in combination with *Z*8 detects sums which are equal to or greater than 6. When sum correction is required for modulo-6 addition, it is performed by the effective subtraction of 6. This is performed by the addition of 1010_2 (10), which is the 2's complement of the BCD representation of 6.



	Shift register				J - K flip-flop	Clock pulse
	A	B	C	MSB D		
Gray input	1	0	1	1	0	Initial state
	0	1	0	1	1	1
	1	0	1	0	0	2
	0	1	0	1	0	3
	0	0	1	0	1	4
Parallel converted output	1	0	0	1	0	5

Serial converted output

Fig. 12.8. Example of conversion for 9.


Addend (000 to 359)
Augend (000 to 359)
200 100
200
100
80 40 20 10
80 40 20 10
8 4 2 1
8 4 2 1
A2 A1 B2 B1
C2 Z5 CO
Σ_2 Σ_1
A4 A3 A2 A1 B4 B3 B2 B1
C4 Z3 CO
Σ_4 Σ_3 Σ_2 Σ_1
A4 A3 A2 A1 B4 B3 B2 B1
C4 Z1 CO
Σ_4 Σ_3 Σ_2 Σ_1
Z9-2
Z8
Z11-2
Z7
Z11-6
Z11-5
Z11-1
Z9-1
Z11-3
Z10-1
Z11-4
Z10-2
A2 A1 B2 B1
N.C.
C2 Z6 CO
Σ_2 Σ_1
10^2
A4 A3 A2 A1 B4 B3 B2 B1
N.C.
C4 Z4 CO
Σ_4 Σ_3 Σ_2 Σ_1
10^1
A4 A3 A2 A1 B4 B3 B2 B1
N.C.
C4 Z2 CO
Σ_4 Σ_3 Σ_2 Σ_1
10^0
Sum (000 to 359)

Components

Z1, 2, 3, 4 SN7483 4-bit adder

Z5, 6 SN7482 2-bit adder

Z7, 8 SN7454 4-wide, 2-input AND-OR-invert gate

Z9 SN7451 Dual, 2-wide, 2-input AND-OR-Invert gates

Z10 SN7400 Quad, 2-input NAND gates

Z11 SN7404 Hex inverters

Fig. 12.9. Three-decade BCD adder, modulo-360.

The hundreds decade performs modulo-4 addition, and thus requires only a 2-bit adder. The gate *Z*9–2 detects the presence of a sum equal to or greater than 3 (disregarding a carry-in), and its output is used to select the proper modulus for the tens decade. A carry-out from the tens decade is applied to a second adder (*Z*6). This allows gate *Z*9–2 to sample the sum of the first adder (*Z*5) without regard for a tens decade carry. The reason for this arrangement is that when the sum of the hundreds digit, disregarding a tens decade carry, gives 200, the tens decade should be set for modulo-10 addition. Now, when a tens carry occurs, if it were entered into the first adder (*Z*5), then the hundreds adder would give an output of 300, and there would be an error because the tens decade would become a modulo-6 adder.

Index

A-c—coupled clocking (*see* Clocking, types of)
A-c noise margin, 41-43
Active pull-up output, 25
Adder, modulo-360, 315-318
Adder/subtracter, BCD, 231
Addition:
  binary, 5-7, 211-212
  binary-coded decimal (BCD), 233
  carry look-ahead, 235-239
Adjacency in Karnaugh map, 111-113
Algorithm:
  for BCD subtracter, 227-228
  development of binary addition, 220-221
Amplifier, two-stage positive-feedback saturating, 161-162
AND circuits, 120-121
  basic operation of, 3-4
AND gates, 51
AND-OR circuits, 127
AND-OR-INVERT (AOI), 52-53
AND-OR-INVERT circuits, 117-118
AND-OR-INVERT gates, 125
Asynchronous controls, 163
Auxiliary functions:
  first level (FLA), 237-238
  second level (SLA), 238

Benedek, 159
Binary counting, 1-2
Binary-decimal equivalents, 2
Binary device (*see* Flip-flops)
Binary number complementation, 154-159
Bistable multivibrator (*see* Flip-flops)
Blanking, 195-196
  displays, 203-208
Boolean algebra:
  postulates and theorems, 107-109
  terminology, 109-110
Boolean ring algebra, 211

Capacitive load:
  effects of, on current spikes, 87-89
  supply current requirements of, 40
Carry, ripple, in addition, 239
Carry generate, 235-239
Carry look-ahead addition, 235
Carry look-ahead in counters, 248-249
Carry propagate, 235-239
Cascading counters, 261
Clear (*see* Asynchronous controls)
Clock-burst generator, 176
Clock skew, 165
Clocking:
  synchronous, 164
  types of, 164
Coaxial lines, noise from, 93-105
Code:
  binary-coded decimal (BCD), 184
    arithmetic, 224-231
  excess-3, 184-185
    arithmetic, 224, 231-235
  excess-3 BCD, 197
  excess-3 decimal, 231-235
Codes:
  common, 181-186
  weighted, 184
Coincidence circuit, 131
Cold-cathode tubes, decoder/driver for, 193-196
Collector-dotting, 127-128
Combining families, 58-59
Comparator, serial, 175
Comparator circuit, 131
Complementation, 114-115
Computers, early, 1
Controls, asynchronous, 163
Conversion between BCD and excess-3, 242
Converter, 131-154
  A-to-*D*, 208-210
  asynchronous, 139
  BCD-to-binary, 149-154
  BCD to excess-3, 135
  binary-to-BCD, 139
  binary-to-Gray, 133-135
  combined binary-to-Gray and Gray-to-Binary, 135
  decimal-to-BCD, 139
  excess-3 to BCD, 135
  Gray-to-binary, 131-133

Converter (*Cont.*)
  parallel BCD to parallel binary, 241
  serial BCD to serial binary, 241–242
  serial binary to parallel BCD, 240–241
Couleur, J. F., 139, 159
Counter:
  binary-coded decimal (BCD), 247
    decade, 254
  divide-by-2, 254
  divide-by-3, 258–259
  divide-by-5, 254
  divide-by-6, 258–259
  divide-by-8, 258–259
  divide-by-10, 254
  divide-by-12, 258
    design procedure, 247
  divide-by-*N*, 271–283
  4-bit binary ripple, 243
  Johnson, 294–298
  maximum length shift, 298–300
  Möbius, 294–298
  nonmaximum length shift, 300–301
  ring, 292
  synchronous: clock frequency, 251–252
    design of, 249–251
    4-bit-binary, 267
    with parallel carry, 261
  twisted ring, 294–298
    Johnson, 189
  up/down: 4-stage binary, 253
    synchronous 4-bit-binary, 261
Counters:
  cascading, 261, 282
  cycle length, 252
  decode spikes in, 244
  ripple: clock frequency of, 244
    resetting, 247–248
Crosstalk, 93–96
Current spiking, I_{CC}, 24

D-c clocking (*see* Clocking, types of)
D-c noise margin, 27, 40–41
Data synchronizer, 176
Datavue® tube, Raytheon, 193
Decimal-to-binary conversion in computers, 1
Decimal-binary equivalents, 2
Decode spikes in counters, 244
Decimal-binary equivalents, 2
Decode spikes in counters, 244
Decoder:
  BCD-to-decimal, 185–186
  seven-segment, 189
  strobing, 186–188, 202
  3-bit binary, 181
Decoder driver:
  BCD-to-7-segment, 197–201
  open-collector, breakdown, 191
Decoder summary chart, 194
Decoupling, 87, 92–93
Decoupling capacitor, calculation of, 89–92

De Falco, John A., 103
Delay element, 55, 57
De Morgan's theorem, 109
  applied, 114–115, 118, 127–128, 131
Demultiplexer, 191, 201–202
Detector, frequency-phase, 180
Diode-transistor logic (DTL), 10–11
Direct-coupled transistor logic (DCTL), 8–9
Displays, decoder drivers for, 202–208
Dot-OR logic, 127–128
Driver, decoder, 189–191
Duality, rule of, 109

Earle, J., 159
Edge-triggered logic (*see* Clocking, types of)
Elmore, W. C., 43
Emiter-base breakdown, 30, 69
Emitter-coupled logic (ECL), 12–13
End-around carry, 224, 229
η, maximum values, 49–51
Excitation table, *J-K*, 168
Exclusive-NOR (comparator) circuits, 131
Exclusive-OR circuits, 128–131
Exclusive-OR operation rules, 131
Expanders, 52–53, 120–121

Fan-out, 25, 57
  calculation of maximum, 49–51
First-level auxiliary functions (FLA), 237–238
Flip-flop:
  D-type, 166
  J-K, 168–170
  master and slave, 164
  R-S, 167–168
  T-type, 166–167
Flip-flop clocking (*see* Clocking, types of)
Flip-flops:
  converting types, 170
  as counters, 243–283
  series 54/74, common characteristics of, 171
Flores, Ivan, 235
Frequency divider, programmable, 261
Frequency divider shift register, 305–307
Frequency division by an integer, 246
Frequency-phase detector, 180
Full-adder, 128–129, 212–214
  basic operation, 6–7
Function table distinguished from truth table, 8

Generator, word, 301–305
Golomb, S. W., 289
Gray code, conversion, 313–315
Ground noise, 92–93
Ground return, 86
Grounding techniques, 92–93
Guaranteed parameters, 25–26

Half-adder, 129, 212
  basic operation of, 6–7
High-speed transistor-transistor logic:
  basic characteristics of, 18–21
  circuits, summary table, 20
Hold time, definition of, 165
Humphrey, W. S., Jr., 159

I_{CC} current spiking, 24
I_{CC} transient peaks, 88
Implementing logic, methods for, 118–131
Input, below ground, 71
Input breakdown, 71, 79
Input breakdown voltages, 69
Input characteristics, 27–30
Input current, multiple emitter load, 58
Input loading, temperature, 78–79
Input terminations, 58
Input/output parameters, guaranteed, 26–35
Integrated-circuit complexity levels, definitions of, 13–14
Inverter circuits, 124–125

JPL Technical Report 32-564, 300

Karnaugh map:
  for BCD, 227
  in binary addition, 212
  for control of seven-segment tube, 189
  in decoding, 181
  described, 111–113
  reduction of terms by, 236
  for synchronous counters, 248–249

Large-scale integration (LSI), 13
Latch:
  D-type, 166
  R-S, 163–164
Latching circuit, 162
Loading rules, 57–65
Logic expression, rules for obtaining, 115–118
Logic functions, definition of, 107–108
Logic gates, 2–4
  basic operation: of AND gate, 3–4
    of NOR gate, 4
    of NOT gate, 4
    of OR gate, 3
Low-power transistor-transistor logic, basic characteristics of, 18
Low-power TTL circuits, summary table, 19

McCluskey, E. J., 159
MacDonald, 159
Maley, G. A., 159
Master-slave clocking (*see* Clocking, types of)
Medium-scale integration (MSI), 13–14
Minimization, 110–113
Modulo-2 adder, 129
Moskowitz, 159
Multi (*see* Flip-flops)
Multiple-emitter input transistor, 24–25
Multiplication, nested, 142–143
Multiplier, binary, 309-312
Mutual coupling, reduction of, 96

NAND circuits, 119–120
NAND gate, basic operation of, 4, 114
  (*See also* Transistor-transistor logic, NAND gate, basic operation of)
Negative logic defined, 8
Nested multiplication, 142–143
Nine's complement, 220
Nixie® tube, Burroughs, 193, 313
Noise:
  effects of, 83
  shielding, 84–85
  sources of, 84
  supply voltage and current, 87–93
Noise coupling, 43–45
Noise rejection, 41–45
Noise susceptibility, 84–85
NOR circuits, 120
NOR gates, 51–52, 114
Normalizing values for fan-out, 57–58
NOT circuit, basic operation, 4
NOT gate, inverter, 114
Notation:
  binary point, 221
  complementary, 221
  sign and magnitude, 221, 235
Notations, binary, comparison of, 224

One's complement, 154
One's complement addition and subtraction, 222
One's complement binary arithmetic, 219–221
One-shots from flip-flops, 178–180
Open-collector gate, 127–128
Open-collector logic, guaranteed ratings, 46–47
OR circuit, basic operation of, 3
OR (inclusive OR) circuits, 123–124
Output breakdown, 79–82
Output characteristics, 30–35
Output drive, temperature, 72–74
Overshoot, 99

Parallel-carry in counters, 248–249
Parameters, guaranteed, 25–26
Positive logic defined, 8
Preset (*see* Asynchronous controls)
Printed circuit boards, noise in, 92–93
Propagation delay:
  of AND-OR invert gate, 52–53
  of flip-flop, 165
  time, 35
Pull-up resistor, 47–51

Quine-McCluskey minimization technique, 113

Radix:
  definition of, 220
  diminished, 220
Radix complement, 220
Redundant variable in Karnaugh map, 113
Register, data selector/storage, 291
Register exchange, 219
Reset (*see* Asynchronous controls)
Resistor-capacitor transistor logic (RCTL), 10
Resistor-transistor logic (RTL), 9–10
Resistor values for combinations of loads and stages, 49–51
Reverse leakage, 68–69
Ringing, 99, 102
Ripple-carry in addition, 239
Ripple-carry in counters, 252
Rise-time sensitivity, 53

Sands, M., 43
Schmitt trigger, 53–57
Schmitt-input NAND gate, 53–57
Schottky barrier diode, 21
Schottky-clamped TTL basic characteristics of, 21
Second-level auxiliary functions (SLA), 238
Series 54/74:
  basic parameters, 25–45
    exceptions to, 45–57
  common characteristics of, 15
  counters, summary table, 254, 257
  gate types, 119–131
  major divisions, 15
  shift registers, summary table, 286
Series 54S/74S, basic characteristics of, 21
Set (*see* Asynchronous controls)
Set-up time:
  definition of, 166
  J-K flip-flop, 169
Seven-segment display, driving, 208
Sheet resistance, 67
Shift register:
  described, 173
  4-bit, 287–291
  5-bit, 289
  8-bit, 286
  parallel-in, series-out, 287
  right-shift left-shift, 287, 289
Shift register generator counter, 298–300
Shift register word generator, 301–305
Sign-bit, 159
Sign and magnitude addition and subtraction, 221
Singleton, Robert S., 103
Sink current, 27
Sklar, 159
Small-scale integration (SSI), 13–14
Source current, 27
Speed-power product summary, illustrated, 13
Standard TTL circuits:
  basic characteristics of, 16–17
  summary table of, 16–17
Stewart, John L., 85
Straight wire lines, noise from, 93–105
Subtraction, BCD, 227-231
Supply current, average per-gate, 36, 38
Supply current requirements, 36–40
Switching speed, guaranteed, 35
Synchronous clocking, 164–165

Temperature variations in components, 69–79
Ten's complement, 220
Testing, 28–43
Thermal characteristics, 69–79
Thermal impedance, 80
Toggle (*see* Flip-flops)
Toggling, 167
Totem-pole output, 25, 45–46, 49, 52
  transient in, 87–88
Transfer characteristics, 23–24, 25–26
Transistor-transistor logic (TTL), 11–12
  advantages of, 24–25
  NAND gate, basic operation of, 23–24
Transmission lines:
  graphical analysis of, 97–103
  noise-coupling effect in, 85–86
  reflections in, 96–102
  rules for minimizing effects in, 105
  selection of, 93–96
Truth table:
  distinguished from function table, 8
  for *D*-type flip-flop, 166
  for *J-K* flip-flop, 168
  for *R-S* flip-flop, 167
  for *T*-type flip-flop, 167
Twisted-pair lines, noise from, 93–105
Two-phase clock generator, 178
Two's complement, 154–159
  in binary arithmetic, 219–221
Two's complement addition and subtraction, 222–224

Unused inputs, 58

V_{CC} voltage spikes, 88–92
Voltage levels, output and threshold, 26

Wickes, W. E., 159
Wire-AND-ing, 45–46
  effect on d-c noise margin, 48
Wired-AND logic, 127–128
Wired-OR logic, 127–128
Worst-case parameters, 32–35

07-063745-8